Fparser Reference Guide

Release 0.0.16

R. W. Ford, P. Peterson, A. R. Porter

Oct 13, 2022

CONTENTS

Welcome to the fparser Reference Guide. This consists of auto-generated API documentation produced using both the *AutoAPI Sphinx plugin* and *Doxygen*.

CHAPTER

ONE

SPHINX AND AUTOAPI-GENERATED DOCUMENTATION

1.1 fparser

1.1.1 Submodules

fparser.api

Public API for Fortran parser.

Module content

fparser.common

Submodules

fparser.common.base_classes

Base classes for all Fortran statement types

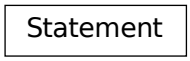
Classes

- *Statement*: Statement instance has attributes:
- *BeginStatement*: [construct_name :] <blocktype> [<name>]
- *EndStatement*: END [<blocktype> [<name>]]
- *Variable*: Variable instance has attributes:
- *AttributeHolder*: Defines a object with predefined attributes. Only those attributes
- *ProgramBlock*: Undocumented.

class fparser.common.base_classes.**Statement**(*parent*, *item*)

Statement instance has attributes:
parent - Parent BeginStatement or FortranParser instance item - Line instance containing the statement line isvalid - boolean, when False, the Statement instance will be ignored

Inheritance



get_provides()
Returns dictonary containing statements that block provides or None when N/A.

get_type(*name*)
Return type declaration using implicit rules for name.

get_variable(*name*)
Return Variable instance of variable name.

class fparser.common.base_classes.**BeginStatement**(*parent*, *item=None*)
[construct_name :] <blocktype> [<name>]

BeginStatement instances have additional attributes:
name blocktype

Block instance has attributes:
content - list of Line or Statement instances name - name of the block, unnamed blocks are named

with the line label

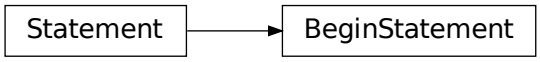
construct_name - name of a construct parent - Block or FortranParser instance item - Line instance containing the block start statement get_item, put_item - methods to retrive/submit Line instances

from/to Fortran reader.

isvalid - boolean, when False, the Block instance will be ignored.

stmt_cls, end_stmt_cls

Inheritance



fill(*end_flag=False*)

Fills blocks content until the end of block statement.

handle_unknown_item_and_raise(*item*)

Called when process_subitem does not find a start or end of block. It adds the item (which is an instance of Line) to the content, but then raises an AnalyzeError. An instance of Line in content typically results in other errors later (e.g. because Line has no analyze method).

process_item()

Process the line

process_subitem(*item*)

Check if item is blocks start statement, if it is, read the block.

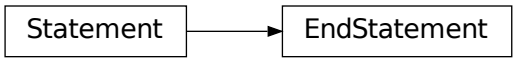
Return True to stop adding items to given block.

class fparser.common.base_classes.**EndStatement**(*parent*, *item*)

END [<blocktype> [<name>]]

EndStatement instances have additional attributes:
name blocktype

Inheritance



tofortran(*isfix=None*)

Returns a valid Fortran string for this END statement. It guarantees that there is no white space after the 'END' in case of an unnamed statement.

Parameters
isfix (*bool*) – True if the code is in fixed format.

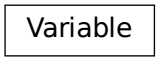
Returns
the (named or unnamed) valid Fortran END statement as a string.

Return type
str

class fparser.common.base_classes.**Variable**(*parent*, *name*)

Variable instance has attributes:
name typedecl dimension attributes intent parent - Statement instances defining the variable

Inheritance



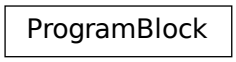
class fparser.common.base_classes.**AttributeHolder**(***kws*)

Defines a object with predefined attributes. Only those attributes are allowed that are specified as keyword arguments of a constructor. When an argument is callable then the corresponding attribute will be read-only and set by the value the callable object returns.

Inheritance

class fparser.common.base_classes.**ProgramBlock**

Inheritance



fparser.common.readfortran

Provides Fortran reader classes.

Overview

Provides FortranReader classes for reading Fortran codes from files and strings. FortranReader handles comments and line continuations of both fix and free format Fortran codes.

Examples

```
>> from fparser.common.readfortran import FortranFileReader
>>> import os
>>> reader = FortranFileReader(os.path.expanduser('~/src/blas/daxpy.f'))
>>> print reader.next()
line #1 'subroutine daxpy(n,da,dx,incx,dy,incy)'
>>> print `reader.next()`
Comment('c     constant times a vector plus a vector.\n
c     uses unrolled loops for increments equal to one.\n
c     jack dongarra, linpack, 3/11/78.\n
c     modified 12/3/93, array(1) declarations changed to array(*)',(3, 6))
>>> print `reader.next()`
Line('double precision dx(*),dy(*),da',(8, 8),'')
>>> print `reader.next()`
Line('integer i,incx,incy,ix,iy,m,mp1,n',(9, 9),'')
```

Note that the `.next()` method may return *Line*, *SyntaxErrorLine*, *Comment*, *MultiLine*, or *SyntaxErrorMultiLine* instance. Let us continue with the above example session to illustrate the *Line* methods and attributes:

```
>>> item = reader.next()
>>> item
    Line('if (da .eq. 0.0d0) return',(12, 12),'')
>>> item.line
    'if (da .eq. 0.0d0) return'
>>> item.strline
    'if (F2PY_EXPR_TUPLE_5) return'
>>> item.strlinemap
    {'F2PY_EXPR_TUPLE_5': 'da .eq. 0.0d0'}
>>> item.span
    (12, 12)
>>> item.get_line()
    'if (F2PY_EXPR_TUPLE_5) return'
```

To read a Fortran code from a string, use *FortranStringReader* class:

```
>>> from fparser.common.sourceinfo import FortranFormat
>>> from fparser.common.readfortran import FortranStringReader
>>> code = '''
...       subroutine foo(a)
...         integer a
...         print*,"a=",a
...       end
... '''
>>> reader = FortranStringReader(code)
>>> reader.set_format(FortranFormat(False, True))
```

(continues on next page)

(continued from previous page)

```
>>> reader.next()
    Line('subroutine foo(a)',(2, 2),'')
>>> reader.next()
    Line('integer a',(3, 3),'')
>>> reader.next()
    Line('print*,"a=",a',(4, 4),'')
```


Classes

- *FortranFileReader*: Constructs a FortranFileReader object from a file.
- *FortranStringReader*: Reads Fortran source code as a string.
- *Line*: Holds a Fortran source line.
- *Comment*: Holds a Fortran comment.
- *MultiLine*: Holds PYF file multiline.

class fparser.common.readfortran.**FortranFileReader**(*file_candidate*, *include_dirs=None*, *source_only=None*, *ignore_comments=True*)

Constructs a FortranFileReader object from a file.

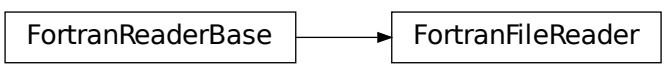
Parameters

- **file_candidate** – A filename or file-like object.
- **include_dirs** (*list*) – Directories in which to look for inclusions.
- **source_only** (*list*) – Fortran source files to search for modules required by "use" statements.
- **ignore_comments** (*bool*) – Whether or not to ignore comments

For example:

```
>>> from fparser.common.readfortran import FortranFileReader
>>> import os
>>> reader = FortranFileReader('myfile.f90')
```

Inheritance



close_source()

Called when self.source.next() raises StopIteration.

class fparser.common.readfortran.**FortranStringReader**(*string*, *include_dirs=None*, *source_only=None*, *ignore_comments=True*)

Reads Fortran source code as a string.

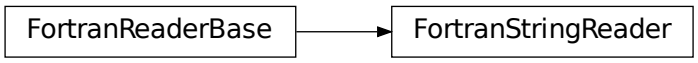
Parameters

- **string** (*str*) – string to read
- **include_dirs** (*list*) – List of dirs to search for include files
- **source_only** (*list*) – Fortran source files to search for modules required by "use" statements.
- **ignore_comments** (*bool*) – Whether or not to ignore comments

For example:

```
>>> from fparser.common.readfortran import FortranStringReader
>>> code = '''
      subroutine foo(a)
         integer a
         print*,"a=",a
       end
    '''
>>> reader = FortranStringReader(code)
```

Inheritance



class fparser.common.readfortran.**Line**(*line*, *linenospan*, *label*, *name*, *reader*)

Holds a Fortran source line.

Attributes

line
[str] code line

span
[2-tuple] starting and ending line numbers

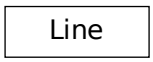
label
[{int, None}] Specify statement label

name
[{str, None}] Specify construct name.

reader : FortranReaderBase strline : {None, str} is_f2py_directive : bool

the line contains f2py directive

Inheritance



apply_map(*line*)

Substitutes magic strings in a line with values specified in a map.

clone(*line*)

This Line has its contents overwitten by the passed string. The incoming string has substitution applied.

copy(*line=None*, *apply_map=False*)

Creates a Line object from a string.

If no line argument is specified a copy is made of this Line.

If a substitution map is provided it is used while making the copy.

has_map()

Returns true when a substitution map has been registered.

class fparser.common.readfortran.**Comment**(*comment*, *linenospan*, *reader*)

Holds a Fortran comment.

Parameters

- **comment** (*str*) – String containing the text of a single or multi-line comment
- **linenospan** (*(int, int)*) – A 2-tuple containing the start and end line numbers of the comment from the input source.
- **reader** (fparser.common.readfortran.FortranReaderBase) – The reader object being used to read the input source.

Inheritance

isempty(*ignore_comments=False*)

Whether or not this comment is in fact empty (or we are ignoring it). Provided for compatibility with Line.isempty()

Parameters
ignore_comments (*bool*) – whether we ignore comments

Returns
True if we are ignoring comments, False otherwise

Return type
bool

class fparser.common.readfortran.**MultiLine**(*prefix*, *block*, *suffix*, *linenospan*, *reader*)

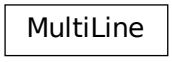
Holds PYF file multiline.

PYF file multiline is represented as follows::
prefix+"'+lines+"'+suffix.

Parameters

- **prefix** (*str*) – the prefix of the line(s)
- **block** (List[*fparser.common.readfortran.Line*]) – list of lines
- **suffix** (*str*) – the suffix of the block of lines
- **linenospan** (*Tuple[int, int]*) – starting and ending line numbers
- **reader** (fparser.common.readfortran.FortranReaderBase) – the current reader instance.

Inheritance



isempty(*ignore_comments=False*)

Returns true if there is no significant text in this multi-line string.

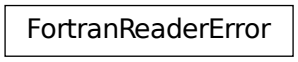
Exceptions

- *FortranReaderError*: Thrown when there is an error reading the Fortran source file.
- *SyntaxErrorLine*: Indicates a syntax error while processing a line.
- *SyntaxErrorMultiLine*: Indicates a syntax error while processing Python multi-line strings.

exception fparser.common.readfortran.**FortranReaderError**

Thrown when there is an error reading the Fortran source file.

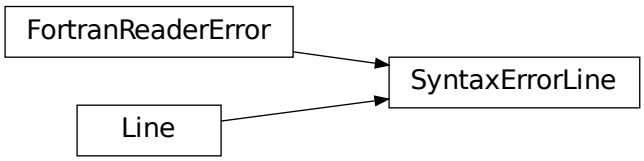
Inheritance



exception fparser.common.readfortran.**SyntaxErrorLine**(*line*, *linenospan*, *label*, *name*, *reader*, *message*)

Indicates a syntax error while processing a line.

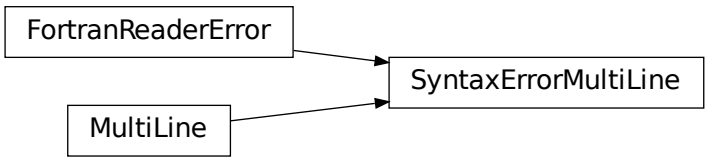
Inheritance



exception fparser.common.readfortran.**SyntaxErrorMultiLine**(*prefix*, *block*, *suffix*, *linenospan*, *reader*, *message*)

Indicates a syntax error while processing Python multi-line strings.

Inheritance



fparser.common.sourceinfo

Provides functions to determine whether a piece of Fortran source is free or fixed format. It also tries to differentiate between strict and "pyf" although I'm not sure what that is.

fparser.common.splitline

Defines LineSplitter and helper functions.

Original Author: Pearu Peterson <pearu@cens.ioc.ee> First version created: May 2006

Functions

- *string_replace_map()*: Undocumented.
- *splitquote()*: Fast LineSplitter
- *splitparen()*: Splits a line into top-level parenthesis and not-parenthesised

fparser.common.splitline.**string_replace_map**(**args*, ***kwargs*)

fparser.common.splitline.**splitquote**(*line*, *stopchar=None*, *lower=False*, *quotechars='"\''*)

Fast LineSplitter

fparser.common.splitline.**splitparen**(*line*, *paren_open='([', paren_close=')]'*)

Splits a line into top-level parenthesis and not-parenthesised parts. E.g.: "a((1+2)*3) = b(x)" becomes: ["a", "((1+2)*3)", " = b", "(x)"] :param str line: the string to split. :param str paren_open: The characters that define an open parentheses. :param str paren_close: The characters that define a closing parentheses. :return: List of parenthesised and not-parenthesised parts :rtype: list of str The paren_open and paren_close strings must be matched in order: paren_open[x] is closed by paren_close[x].

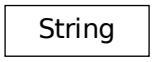
Classes

- *String*: Dummy string class.

class fparser.common.splitline.**String**

Dummy string class.

Inheritance



fparser.common.tests

Submodules

fparser.common.tests.logging_utils

Helps with testing methods which write to the standard logger.

fparser.common.utils

Various utility functions.

Permission to use, modify, and distribute this software is given under the terms of the NumPy License. See http://scipy.org.

NO WARRANTY IS EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK. Author: Pearu Peterson <pearu@cens.ioc.ee> Created: May 2006

Functions

- *split_comma()*: Split (an optionally bracketed) comma-separated list into
- *specs_split_comma()*: Undocumented.
- *get_module_file()*: Undocumented.
- *parse_bind()*: Undocumented.
- *parse_result()*: Undocumented.

- *parse_array_spec()*: Undocumented.
- *str2stmt()*: Convert Fortran code to Statement tree.

fparser.common.utils.**split_comma**(*line*, *item=None*, *comma=','*, *keep_empty=False*, *brackets=None*)

Split (an optionally bracketed) comma-separated list into items and return a list containing them. If supplied then brackets must be a list of containing two strings, the first being the opening bracket and the second the closing bracket.

fparser.common.utils.**specs_split_comma**(*line*, *item=None*, *upper=False*)

fparser.common.utils.**get_module_file**(*name*, *directory*, *_cache={}*)

fparser.common.utils.**parse_bind**(*line*, *item=None*)

fparser.common.utils.**parse_result**(*line*, *item=None*)

fparser.common.utils.**parse_array_spec**(*line*, *item=None*)

fparser.common.utils.**str2stmt**(*string*, *isfree=True*, *isstrict=False*)

Convert Fortran code to Statement tree.

Classes

- *classes*: Make classes available as attributes of this class.

class fparser.common.utils.**classes**(*name*, *bases*, *dict*)

Make classes available as attributes of this class.

To add a class to the attributes list, one must use:

```
class Name(metaclass=classes):
```

in the definition of the class.

In addition, apply the following tasks:

- decorate analyze methods with show_item_on_failure

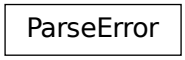
Inheritance

Exceptions

- *ParseError*: Common base class for all non-exit exceptions.
- *AnalyzeError*: Common base class for all non-exit exceptions.

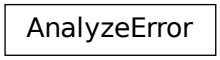
exception fparser.common.utils.**ParseError**

Inheritance



exception fparser.common.utils.**AnalyzeError**

Inheritance



Variables

- *is_name*

fparser.common.utils.**is_name**

Matches zero or more characters at the beginning of the string.

```
<built-in method match of re.Pattern object at 0x5582489511b0>
```

fparser.one

Submodules

fparser.one.block_statements

Fortran block statements.

Classes

- *BeginSource*: Fortran source content.
- *Module*: MODULE <name>
- *PythonModule*: PYTHON MODULE <name>
- *Program*: PROGRAM [name]
- *BlockData*: BLOCK DATA [<block-data-name>]
- *Interface*: INTERFACE [<generic-spec>] | ABSTRACT INTERFACE
- *Subroutine*: [<prefix>] SUBROUTINE <name> [([<dummy-arg-list>])
- *Function*: [<prefix>] FUNCTION <name> ([<dummy-arg-list>]) [<suffix>]
- *SelectCase*: [<case-construct-name> :] SELECT CASE (<case-expr>)
- *SelectType*: [<case-construct-name> :] SELECT TYPE (<case-expr>)
- *WhereConstruct*: [<where-construct-name> :] WHERE (<mask-expr>)
- *ForallConstruct*: [<forall-construct-name> :] FORALL <forall-header>
- *IfThen*: [<if-construct-name> :] IF (<scalar-logical-expr>) THEN
- *If*: IF (<scalar-logical-expr>) action-stmt
- *Do*: [<do-construct-name> :] DO label [loopcontrol]
- *Associate*: [<associate-construct-name> :] ASSOCIATE (<association-list>)
- *TypeDecl*: TYPE [[, <typ-attr-spec-list>] ::] <type-name> [(<type-param-name-list>)]
- *Enum*: ENUM , BIND(C)
- *EndSource*: Dummy End statement for BeginSource.
- *EndModule*: END [<blocktype> [<name>]]
- *EndPythonModule*: END [<blocktype> [<name>]]
- *EndProgram*: END [PROGRAM [name]]

- *EndBlockData*: END [BLOCK DATA [<block-data-name>]]
- *EndInterface*: END [<blocktype> [<name>]]
- *EndSubroutine*: END [SUBROUTINE [name]]
- *EndFunction*: END [FUNCTION [name]]
- *EndSelect*: END [<blocktype> [<name>]]
- *EndWhere*: END WHERE [<where-construct-name>]
- *EndForall*: END FORALL [<forall-construct-name>]
- *EndIfThen*: END IF [<if-construct-name>]
- *EndDo*: END DO [<do-construct-name>]
- *EndAssociate*: END ASSOCIATE [<associate-construct-name>]
- *EndType*: END TYPE [<type-name>]
- *EndEnum*: END ENUM
- *GeneralAssignment*: <variable> = <expr>
- *Assignment*: <variable> = <expr>
- *PointerAssignment*: <variable> = <expr>
- *Assign*: ASSIGN <label> TO <int-variable-name>
- *Call*: Call statement class
- *Goto*: GO TO <label>
- *ComputedGoto*: GO TO (<label-list>) [,] <scalar-int-expr>
- *AssignedGoto*: GO TO <int-variable-name> [(<label> [, <label>]...)]
- *Continue*: CONTINUE
- *Return*: RETURN [<scalar-int-expr>]
- *Stop*: STOP [<stop-code>]
- *Print*: PRINT <format> [, <output-item-list>]
- *Read*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Read0*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Read1*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Write*: WRITE (io-control-spec-list) [<output-item-list>]
- *Flush*: FLUSH <file-unit-number>
- *Wait*: WAIT (<wait-spec-list>)
- *Contains*: CONTAINS
- *Allocate*: ALLOCATE ([<type-spec> ::] <allocation-list> [, <alloc-opt-list>])
- *Deallocate*: DEALLOCATE (<allocate-object-list> [, <dealloc-opt-list>])
- *ModuleProcedure*: [MODULE] PROCEDURE [::] <procedure-name-list>
- *Access*: <access-spec> [[::] <access-id-list>]
- *Public*: <access-spec> [[::] <access-id-list>]

- *Private*: <access-spec> [[::] <access-id-list>]
- *Close*: CLOSE (<close-spec-list>)
- *Cycle*: CYCLE [<do-construct-name>]
- *Backspace*: REWIND <file-unit-number>
- *Endfile*: REWIND <file-unit-number>
- *Rewind*: REWIND <file-unit-number>
- *Open*: OPEN (<connect-spec-list>)
- *Format*: FORMAT <format-specification>
- *Save*: SAVE [[::] <saved-entity-list>]
- *Data*: DATA <data-stmt-set> [[,] <data-stmt-set>]...
- *Nullify*: NULLIFY (<pointer-object-list>)
- *Use*: Parses USE statement.
- *Exit*: EXIT [<do-construct-name>]
- *Parameter*: PARAMETER (<named-constant-def-list>)
- *Equivalence*: EQUIVALENCE <equivalence-set-list>
- *Dimension*: DIMENSION [::] <array-name> (<array-spec>)
- *Target*: TARGET [::] <object-name> (<array-spec>)
- *Pointer*: POINTER [::] <pointer-decl-list>
- *Protected*: PROTECTED [::] <entity-name-list>
- *Volatile*: VOLATILE [::] <object-name-list>
- *Value*: VALUE [::] <dummy-arg-name-list>
- *ArithmeticIf*: IF (<scalar-numeric-expr>) <label> , <label> , <label>
- *Intrinsic*: INTRINSIC [::] <intrinsic-procedure-name-list>
- *Inquire*: INQUIRE (<inquire-spec-list>)
- *Sequence*: SEQUENCE
- *External*: EXTERNAL [::] <external-name-list>
- *Namelist*: NAMELIST / <namelist-group-name> / <namelist-group-object-list> [[,]
- *Common*: COMMON [/ [<common-block-name>] /] <common-block-object-list> [[,] / [<common-block-name>] / <common-block-object-list>]...
- *Optional*: OPTIONAL [::] <dummy-arg-name-list>
- *Intent*: INTENT (<intent-spec>) [::] <dummy-arg-name-list>
- *Entry*: ENTRY <entry-name> [([<dummy-arg-list>]) [<suffix>]]
- *Import*: IMPORT [[::] <import-name-list>]
- *ForallStmt*: FORALL <forall-header> <forall-assignment-stmt>
- *SpecificBinding*: PROCEDURE [(<interface-name>)] [[, <binding-attr-list>] ::]
- *GenericBinding*: GENERIC [, <access-spec>] :: <generic-spec> => <binding-name-list>

- *FinalBinding*: FINAL [::] <final-subroutine-name-list>
- *Allocatable*: ALLOCATABLE [::] <object-name> [(<deferred-shape-spec-list>)]
- *Asynchronous*: ASYNCHRONOUS [::] <object-name-list>
- *Bind*: <language-binding-spec> [::] <bind-entity-list>
- *Else*: ELSE [<if-construct-name>]
- *ElseIf*: ELSE IF (<scalar-logical-expr>) THEN [<if-construct-name>]
- *Case*: CASE <case-selector> [<case-construct-name>]
- *TypeIs*: TYPE IS <type-selector> [<case-construct-name>]
- *ClassIs*: CLASS <class-selector>
- *WhereStmt*: WHERE (<mask-expr>) <where-assignment-stmt>
- *ElseWhere*: ELSE WHERE (<mask-expr>) [<where-construct-name>]
- *Enumerator*: ENUMERATOR [::] <enumerator-list>
- *FortranName*: FORTRANNAME <name>
- *Threadsafe*: THREADSAFE
- *Depend*: DEPEND (<name-list>) [::] <dummy-arg-name-list>
- *Check*: CHECK (<c-int-scalar-expr>) [::] <name>
- *CallStatement*: CALLSTATEMENT <c-expr>
- *CallProtoArgument*: CALLPROTOARGUMENT <c-type-spec-list>
- *Pause*: PAUSE [<char-literal-constant|int-literal-constant>]
- *Comment*: Attributes
- *Integer*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Real*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *DoublePrecision*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Complex*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *DoubleComplex*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Character*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Logical*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Byte*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *TypeStmt*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Class*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Implicit*: IMPLICIT <implicit-spec-list>

class fparser.one.block_statements.**BeginSource**(*parent*, *item=None*)

Fortran source content.

Inheritance

end_stmt_cls

alias of *EndSource*

process_item()

Process the line

process_subitem(*item*)

Check if item is blocks start statement, if it is, read the block.

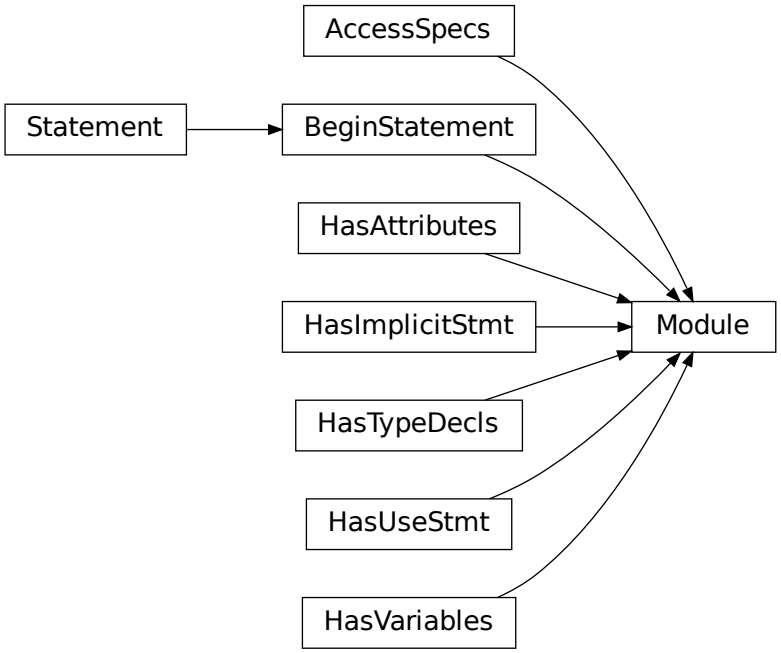
Return True to stop adding items to given block.

class fparser.one.block_statements.**Module**(*parent*, *item=None*)

MODULE <name>

END [MODULE [name]]

Inheritance



end_stmt_cls

alias of *EndModule*

get_provides()

Returns dictonary containing statements that block provides or None when N/A.

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

topyf(*tab=''*)

Constructs a pyf representation of this class.

Parameters
tab (*str*) – White space to prepend to output.

Returns
pyf code for this implicit statement.

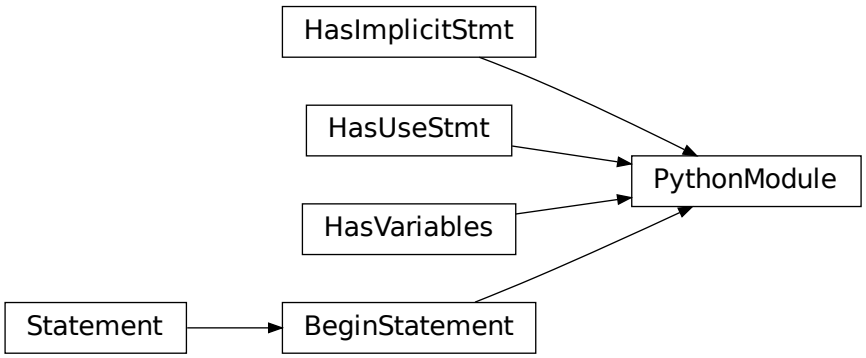
Return type
str

class fparser.one.block_statements.**PythonModule**(*parent*, *item=None*)

PYTHON MODULE <name>

END [PYTHON MODULE [name]]

Inheritance



end_stmt_cls

alias of *EndPythonModule*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

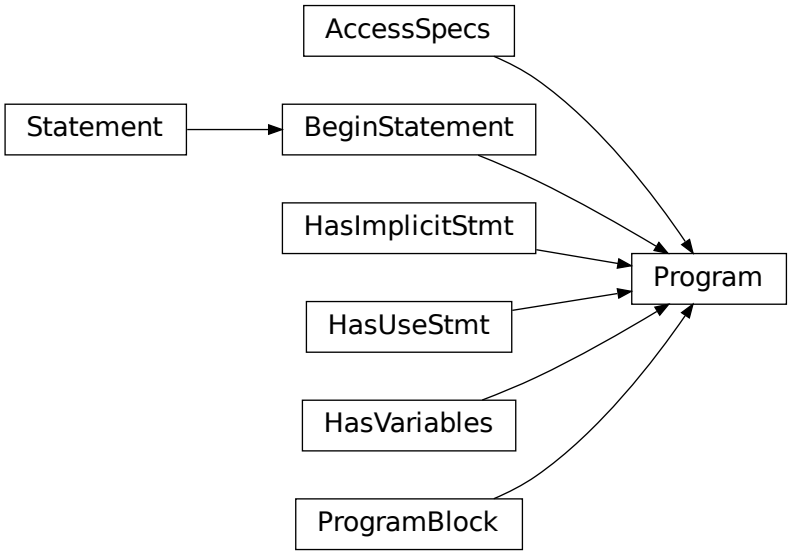
process_item()

Process the line

class fparser.one.block_statements.**Program**(*parent*, *item=None*)

PROGRAM [name]

Inheritance



end_stmt_cls

alias of *EndProgram*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

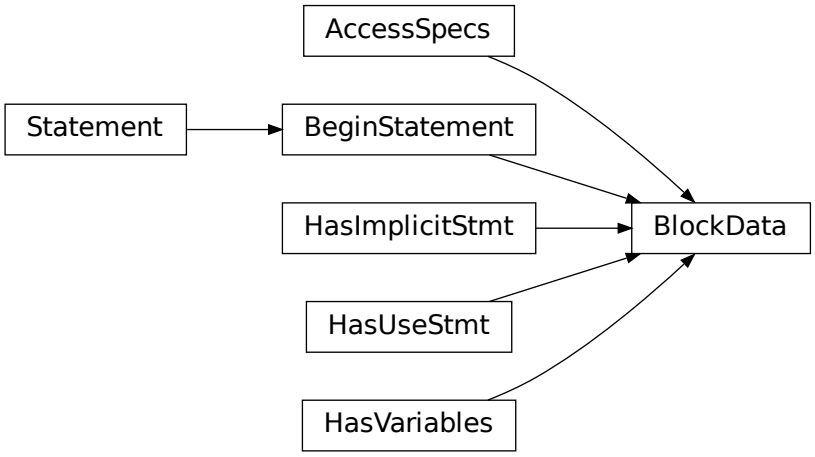
process_item()

Process the line

class fparser.one.block_statements.**BlockData**(*parent*, *item=None*)

BLOCK DATA [<block-data-name>]

Inheritance



end_stmt_cls

alias of *EndBlockData*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

class fparser.one.block_statements.**Interface**(*parent*, *item=None*)

INTERFACE [<generic-spec>] | ABSTRACT INTERFACE END INTERFACE [<generic-spec>]

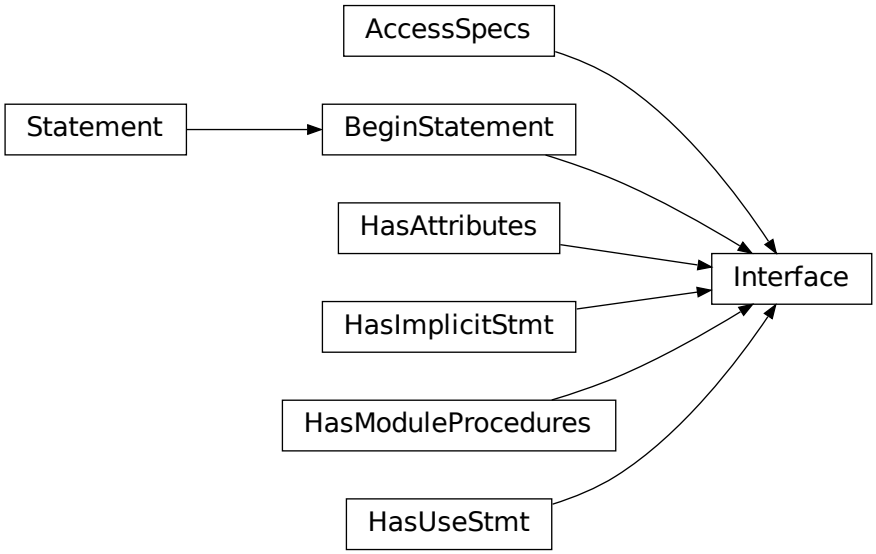
<generic-spec> = <generic-name>

OPERATOR (<defined-operator>)
ASSIGNMENT (=)
<dtio-generic-spec>

<dtio-generic-spec> = READ (FORMATTED)

READ (UNFORMATTED)
WRITE (FORMATTED)
WRITE (UNFORMATTED)

Inheritance



end_stmt_cls

alias of *EndInterface*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

topyf(*tab=''*)

Constructs a pyf representation of this class.

Parameters
tab (*str*) – White space to prepend to output.

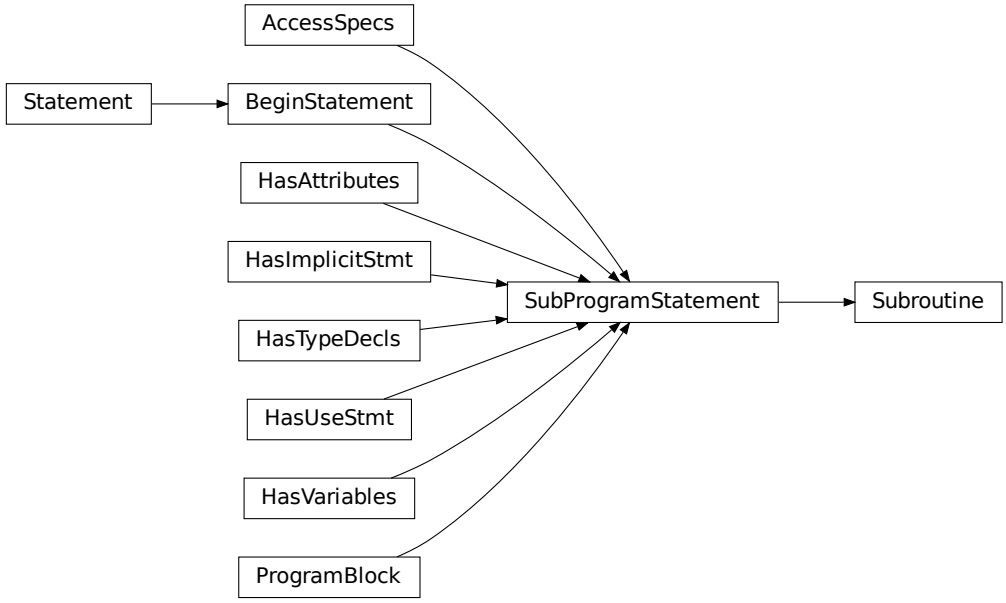
Returns
pyf code for this implicit statement.

Return type
str

class fparser.one.block_statements.**Subroutine**(*parent*, *item=None*)

[<prefix>] SUBROUTINE <name> [([<dummy-arg-list>]) [<proc-language-binding-spec>]]

Inheritance



end_stmt_cls

alias of *EndSubroutine*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Function**(*parent*, *item=None*)

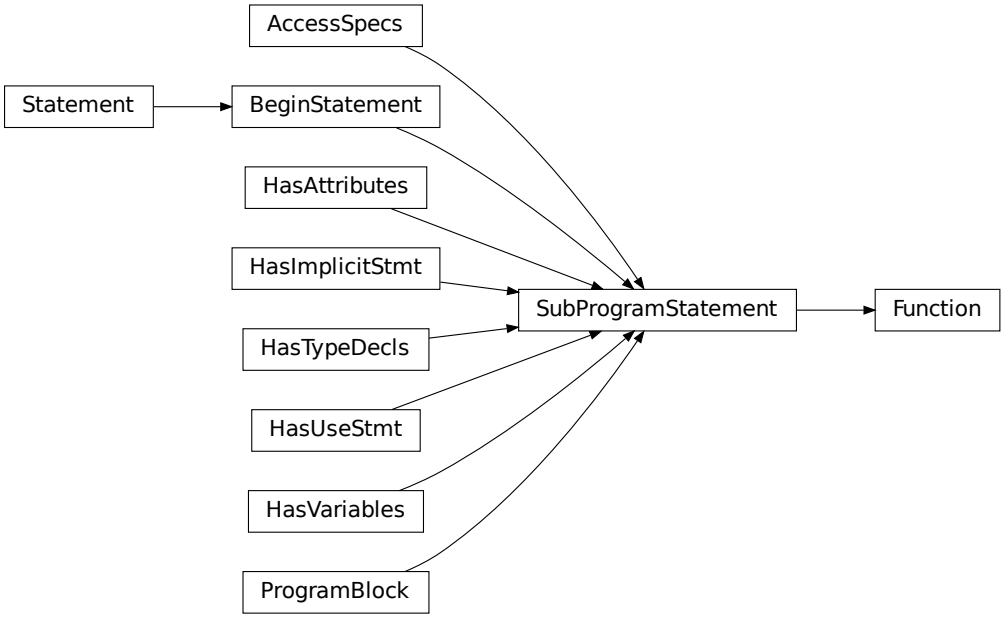
[<prefix>] FUNCTION <name> ([<dummy-arg-list>]) [<suffix>] <prefix> = <prefix-spec> [<prefix-spec>]… <prefix-spec> = <declaration-type-spec>

RECURSIVE | PURE | ELEMENTAL

<suffix> = <proc-language-binding-spec> [RESULT (<result-name>)]

RESULT (<result-name>) [<proc-language-binding-spec>]

Inheritance



end_stmt_cls

alias of *EndFunction*

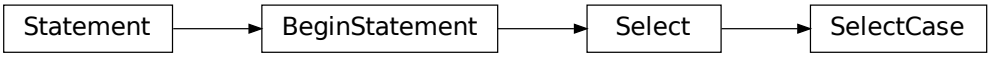
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**SelectCase**(*parent*, *item=None*)

[<case-construct-name> :] SELECT CASE (<case-expr>)

Inheritance



get_classes()

Return the list of classes that this instance may have as children

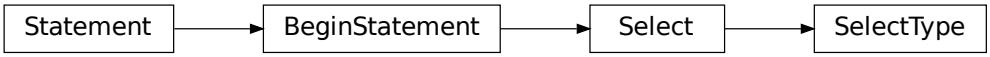
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**SelectType**(*parent*, *item=None*)

[<case-construct-name> :] SELECT TYPE (<case-expr>)

Inheritance



get_classes()

Return the list of classes that this instance may have as children

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

fparser.one.block_statements.**WhereConstruct**

alias of Where

fparser.one.block_statements.**ForallConstruct**

alias of Forall

class fparser.one.block_statements.**IfThen**(*parent*, *item=None*)

[<if-construct-name> :] IF (<scalar-logical-expr>) THEN

IfThen instance has the following attributes:

expr

Inheritance

end_stmt_cls

alias of *EndIfThen*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

class fparser.one.block_statements.**If**(*parent*, *item=None*)

IF (<scalar-logical-expr>) action-stmt

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

class fparser.one.block_statements.**Do**(*parent*, *item=None*)

[<do-construct-name> :] DO label [loopcontrol] [<do-construct-name> :] DO [loopcontrol]

Inheritance

end_stmt_cls

alias of *EndDo*

item_re(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Parses the next line assuming it is a “Do” statement.

Overrides method in *BeginStatement*.

process_subitem(*item*)

Check if item is blocks start statement, if it is, read the block.

Return True to stop adding items to given block.

class fparser.one.block_statements.**Associate**(*parent*, *item=None*)

[<associate-construct-name> :] ASSOCIATE (<association-list>)
<block>

<association> = <associate-name> => <selector> <selector> = <expr> | <variable>

Inheritance

end_stmt_cls

alias of *EndAssociate*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

fparser.one.block_statements.**TypeDecl**

alias of Type

class fparser.one.block_statements.**Enum**(*parent*, *item=None*)

ENUM , BIND(C)
<enumerator-def-stmt> [<enumerator-def-stmt>]…

Inheritance

end_stmt_cls

alias of *EndEnum*

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the line

class fparser.one.block_statements.**EndSource**(*parent*, *item*)

Dummy End statement for BeginSource.

Inheritance

class fparser.one.block_statements.**EndModule**(*parent*, *item*)

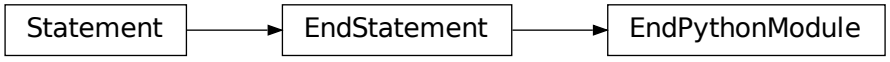
Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndPythonModule**(*parent*, *item*)

Inheritance



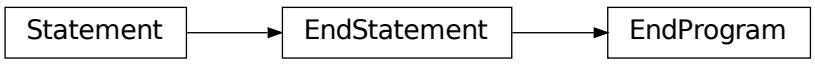
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndProgram**(*parent*, *item*)

END [PROGRAM [name]]

Inheritance



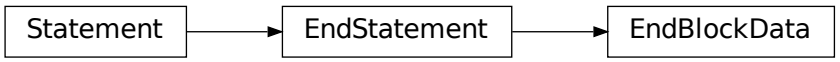
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndBlockData**(*parent*, *item*)

END [BLOCK DATA [<block-data-name>]]

Inheritance

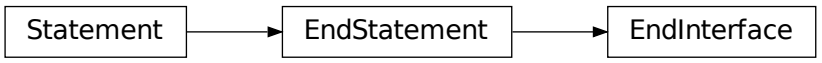


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndInterface**(*parent*, *item*)

Inheritance



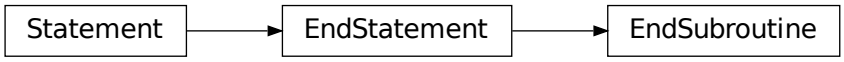
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndSubroutine**(*parent*, *item*)

END [SUBROUTINE [name]]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndFunction**(*parent*, *item*)

END [FUNCTION [name]]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndSelect**(*parent*, *item*)

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndWhere**(*parent*, *item*)

END WHERE [<where-construct-name>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndForall**(*parent*, *item*)

END FORALL [<forall-construct-name>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndIfThen**(*parent*, *item*)

END IF [<if-construct-name>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndDo**(*parent*, *item*)

END DO [<do-construct-name>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

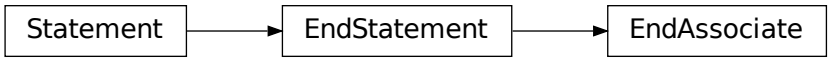
Parses the next line assuming it is an "End do" statement.

Overrides method in *EndStatement*.

class fparser.one.block_statements.**EndAssociate**(*parent*, *item*)

END ASSOCIATE [<associate-construct-name>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndType**(*parent*, *item*)

END TYPE [<type-name>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**EndEnum**(*parent*, *item*)

END ENUM

Inheritance

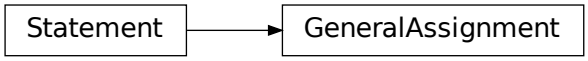
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**GeneralAssignment**(*parent*, *item*)

<variable> = <expr> <pointer variable> => <expr>

Inheritance



item_re(*pos=0*, *endpos=9223372036854775807*)

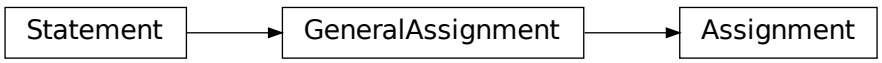
Matches zero or more characters at the beginning of the string.

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

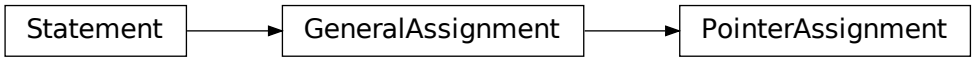
class fparser.one.block_statements.**Assignment**(*parent*, *item*)

Inheritance



class fparser.one.block_statements.**PointerAssignment**(*parent*, *item*)

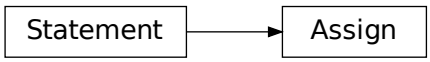
Inheritance



class fparser.one.block_statements.**Assign**(*parent*, *item*)

ASSIGN <label> TO <int-variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Call**(*parent*, *item*)

Call statement class CALL <procedure-designator> [([<actual-arg-spec-list>])]

<procedure-designator> = <procedure-name>

<proc-component-ref>
<data-ref> % <binding-name>

<actual-arg-spec> = [<keyword> =] <actual-arg> <actual-arg> = <expr>

<variable>
<procedure-name>
<proc-component-ref>
<alt-return-spec>

<alt-return-spec> = * <label>

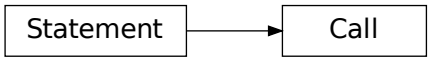
<proc-component-ref> = <variable> % <procedure-component-name>

<variable> = <designator>

Call instance has attributes:

designator arg_list

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Parse the string containing the Call and store the designator and list of arguments (if any)

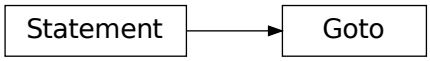
tofortran(*isfix=None*)

Returns the Fortran representation of this object as a string

class fparser.one.block_statements.**Goto**(*parent*, *item*)

GO TO <label>

Inheritance



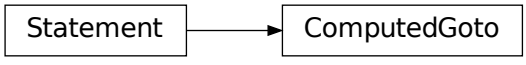
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**ComputedGoto**(*parent*, *item*)

GO TO (<label-list>) [,] <scalar-int-expr>

Inheritance



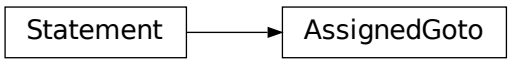
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**AssignedGoto**(*parent*, *item*)

GO TO <int-variable-name> [(<label> [, <label>]...)]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Continue**(*parent*, *item*)

CONTINUE

Inheritance

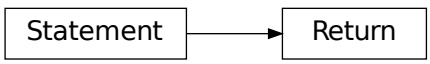
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Return**(*parent*, *item*)

RETURN [<scalar-int-expr>]

Inheritance



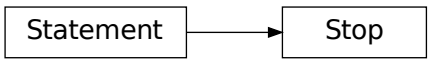
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Stop**(*parent*, *item*)

STOP [<stop-code>] <stop-code> = <scalar-char-constant> | <1-5-digit>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Print**(*parent*, *item*)

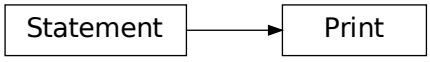
PRINT <format> [, <output-item-list>] <format> = <default-char-expr> | <label> | *

<output-item> = <expr> | <io-implied-do> <io-implied-do> = (<io-implied-do-object-list> , <implied-do-control>) <io-implied-do-object> = <input-item> | <output-item> <implied-do-control> = <do-variable>

= <scalar-int-expr> ,
<scalar-int-expr> [, <scalar-int-expr>]

<input-item> = <variable> | <io-implied-do>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Read**(*parent*, *item*)

Read0: READ (<io-control-spec-list>) [<input-item-list>]

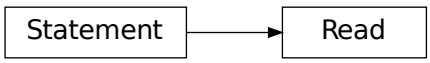
<io-control-spec-list> = [UNIT =] <io-unit>

[FORMAT =] <format>
[NML =] <namelist-group-name>
ADVANCE = <scalar-default-char-expr>

...

Read1: READ <format> [, <input-item-list>]
<format> == <default-char-expr> | <label> | *

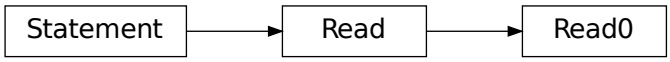
Inheritance



match(*pos=0*, *endpos=9223372036854775807*)
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Read0**(*parent*, *item*)

Inheritance

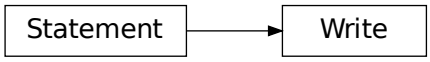


class fparser.one.block_statements.**Read1**(*parent*, *item*)

Inheritance

class fparser.one.block_statements.**Write**(*parent*, *item*)
WRITE (io-control-spec-list) [<output-item-list>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

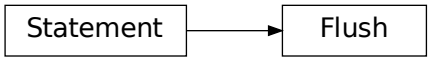
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Flush**(*parent*, *item*)

FLUSH <file-unit-number> FLUSH (<flush-spec-list>) <flush-spec> = [UNIT =] <file-unit-number>

IOSTAT = <scalar-int-variable>
IOMSG = <iomsg-variable>
ERR = <label>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

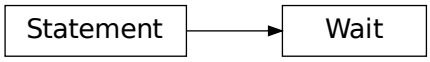
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Wait**(*parent*, *item*)

WAIT (<wait-spec-list>) <wait-spec> = [UNIT =] <file-unit-number>

END = <label>
EOR = <label>
ERR = <label>
ID = <scalar-int-expr>
IOMSG = <iomsg-variable>
IOSTAT = <scalar-int-variable>

Inheritance



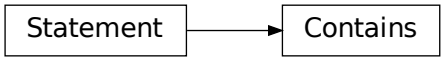
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Contains**(*parent*, *item*)

CONTAINS

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

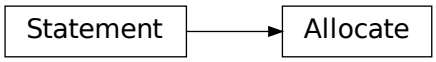
class fparser.one.block_statements.**Allocate**(*parent*, *item*)

ALLOCATE ([<type-spec> ::] <allocation-list> [, <alloc-opt-list>]) <alloc-opt> = STAT = <stat-variable>

ERRMSG = <errmsg-variable>
SOURCE = <source-expr>

<allocation> = <allocate-object> [(<allocate-shape-spec-list>)]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the ALLOCATE statement and store the various entities being allocated in self.items. Any type-specification is stored in self.spec.

Raises
ParseError – if an invalid type-specification is used

tofortran(*isfix=None*)

Create the Fortran code for this ALLOCATE statement

Parameters
isfix (*bool*) – whether or not to generate fixed-format code

Returns
Fortran code

Return type
str

class fparser.one.block_statements.**Deallocate**(*parent*, *item*)

DEALLOCATE (<allocate-object-list> [, <dealloc-opt-list>]) <allocate-object> = <variable-name>

<structure-component>

<structure-component> = <data-ref> <dealloc-opt> = STAT = <stat-variable>

ERRMSG = <errmsg-variable>

Inheritance

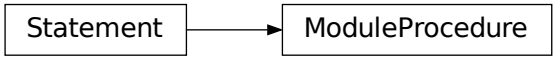
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**ModuleProcedure**(*parent*, *item*)

[MODULE] PROCEDURE [::] <procedure-name-list>

Inheritance



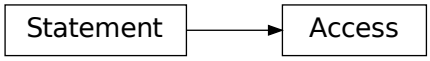
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Access**(*parent*, *item*)

<access-spec> [[::] <access-id-list>] <access-spec> = PUBLIC | PRIVATE <access-id> = <use-name> | <generic-spec>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Public**(*parent*, *item*)

Inheritance

class fparser.one.block_statements.**Private**(*parent*, *item*)

Inheritance

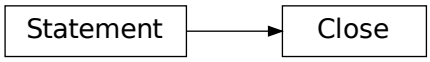
class fparser.one.block_statements.**Close**(*parent*, *item*)

CLOSE (<close-spec-list>) <close-spec> = [UNIT =] <file-unit-number>

IOSTAT = <scalar-int-variable>

IOMSG = <iomsg-variable>
ERR = <label>
STATUS = <scalar-default-char-expr>

Inheritance



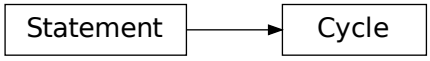
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Cycle**(*parent*, *item*)

CYCLE [<do-construct-name>]

Inheritance

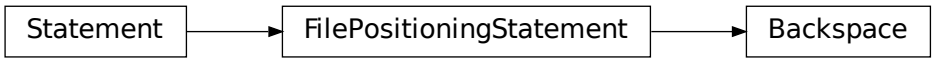


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

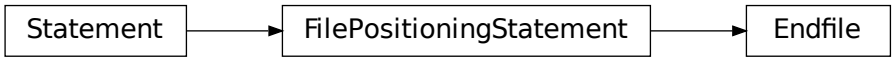
class fparser.one.block_statements.**Backspace**(*parent*, *item*)

Inheritance



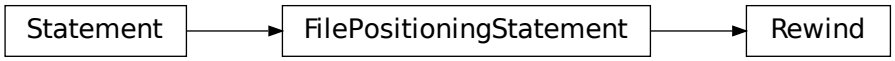
class fparser.one.block_statements.**Endfile**(*parent*, *item*)

Inheritance



class fparser.one.block_statements.**Rewind**(*parent*, *item*)

Inheritance



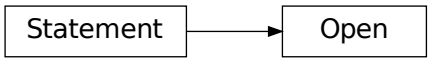
class fparser.one.block_statements.**Open**(*parent*, *item*)

OPEN (<connect-spec-list>) <connect-spec> = [UNIT =] <file-unit-number>

ACCESS = <scalar-default-char-expr>

..

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Format**(*parent*, *item*)

FORMAT <format-specification> <format-specification> = ([<format-item-list>]) <format-item> = [<r>] <data-edit-descr>

<control-edit-descr>

<char-string-edit-descr>

[<r>] (<format-item-list>)

<data-edit-descr> = I <w> [. <m>]

B <w> [. <m>]

...

<r|w|m|d|e> = <int-literal-constant> <v> = <signed-int-literal-constant> <control-edit-descr> = <position-edit-descr>

[<r>] /
:

...

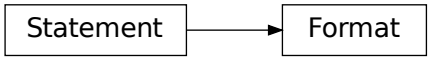
<position-edit-descr> = T <n>

TL <n>

...

<sign-edit-descr> = SS | SP | S ...

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

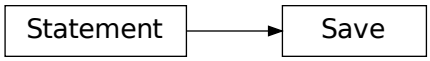
class fparser.one.block_statements.**Save**(*parent*, *item*)

SAVE [[::] <saved-entity-list>] <saved-entity> = <object-name>

<proc-pointer-name>
/ <common-block-name> /

<proc-pointer-name> = <name> <object-name> = <name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Data**(*parent*, *item*)

DATA <data-stmt-set> [[,] <data-stmt-set>]... <data-stmt-set> = <data-stmt-object-list> / <data-stmt-value-list> / <data-stmt-object> = <variable> | <data-implied-do> <data-implied-do> = (<data-i-do-object-list> ,

<data-i-do-variable> = <scalar-int-expr> , <scalar-int-expr> [, <scalar-int-expr>])

<data-i-do-object> = <array-element>

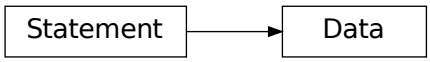
<scalar-structure-component>
<data-implied-do>

<data-i-do-variable> = <scalar-int-variable> <variable> = <designator> <designator> = <object-name>

<array-element>
<array-section>
<structure-component>
<substring>

<array-element> = <data-ref> <array-section> = <data-ref> [(<substring-range>)]

Inheritance



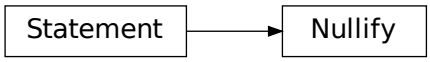
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Nullify**(*parent*, *item*)

NULLIFY (<pointer-object-list>) <pointer-object> = <variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Use**(*parent*, *item*)

Parses USE statement.

Parameters
Statement (*class*) – Base fparser class.

Raises
AnalyzeError – If entity name is not in module.

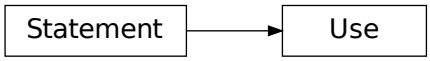
Fortran syntax construct:

USE [[, <module-nature>] ::] <module-name> [, <rename-list>] USE [[, <module-nature>] ::] <module-name> , ONLY : [<only-list>] <module-nature> = INTRINSIC | NON_INTRINSIC <rename> = <local-name> => <use-name>

OPERATOR (<local-defined-operator>) => OPERATOR (<use-defined-operator>)

<only> = <generic-spec> | <only-use-name> | <rename> <only-use-name> = <use-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

populate_use_provides(*all_mod_provides*, *use_provides*, *name*, *rename=None*)

Checks for entity name in the module

process_item()

Parse the string containing the Use and store the module name and list of attributes (if any)

tofortran(*isfix=None*)

Returns the Fortran representation of this object as a string

Parameters
isfix (*bool*) – Whether or not to generated fixed-format Fortran

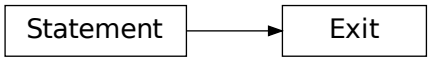
Returns
Fortran representation of this object

Return type
str

class fparser.one.block_statements.**Exit**(*parent*, *item*)

EXIT [<do-construct-name>]

Inheritance



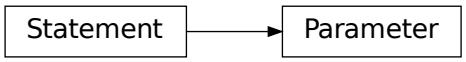
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Parameter**(*parent*, *item*)

PARAMETER (<named-constant-def-list>) <named-constant-def> = <named-constant> = <initialization-expr>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Equivalence**(*parent*, *item*)

EQUIVALENCE <equivalence-set-list> <equivalence-set> = (<equivalence-object> , <equivalence-object-list>) <equivalence-object> = <variable-name> | <array-element> | <substring>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

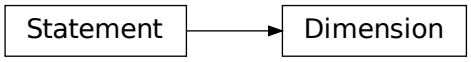
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Dimension**(*parent*, *item*)

DIMENSION [::] <array-name> (<array-spec>)

[, <array-name> (<array-spec>)]...

Inheritance



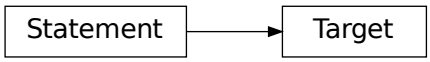
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Target**(*parent*, *item*)

TARGET [::] <object-name> (<array-spec>)
[, <object-name> (<array-spec>)]...

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

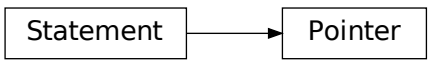
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Pointer**(*parent*, *item*)

POINTER [::] <pointer-decl-list> <pointer-decl> = <object-name> [(<deferred-shape-spec-list>)]

<proc-entity-name>

Inheritance



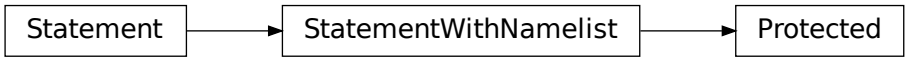
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Protected**(*parent*, *item*)

PROTECTED [::] <entity-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Volatile**(*parent*, *item*)

VOLATILE [::] <object-name-list>

Inheritance

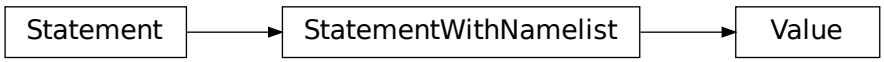
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Value**(*parent*, *item*)

VALUE [::] <dummy-arg-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**ArithmeticIf**(*parent*, *item*)

IF (<scalar-numeric-expr>) <label> , <label> , <label>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Intrinsic**(*parent*, *item*)

INTRINSIC [::] <intrinsic-procedure-name-list>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Inquire**(*parent*, *item*)

INQUIRE (<inquire-spec-list>) INQUIRE (IOLENGTH = <scalar-int-variable>) <output-item-list>

<inquire-spec> = [UNIT =] <file-unit-number>

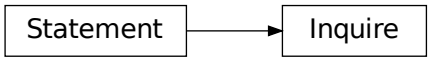
FILE = <file-name-expr>

...

<output-item> = <expr>

<io-implied-do>

Inheritance



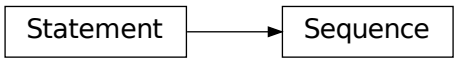
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Sequence**(*parent*, *item*)

SEQUENCE

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**External**(*parent*, *item*)

EXTERNAL [::] <external-name-list>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

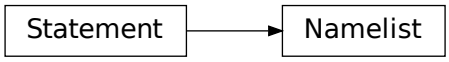
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Namelist**(*parent*, *item*)

NAMELIST / <namelist-group-name> / <namelist-group-object-list> [[,]
/ <namelist-group-name> / <namelist-group-object-list>]...

<namelist-group-object> = <variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

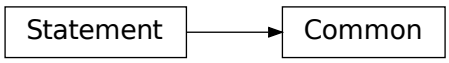
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Common**(*parent*, *item*)

COMMON [/ [<common-block-name>] /] <common-block-object-list> [[,] / [<common-block-name>] / <common-block-object-list>]... <common-block-object> = <variable-name> [(<explicit-shape-spec-list>)]

<proc-pointer-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Optional**(*parent*, *item*)

OPTIONAL [::] <dummy-arg-name-list> <dummy-arg-name> = <name>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

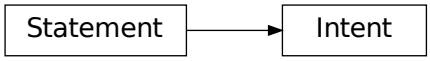
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Intent**(*parent*, *item*)

INTENT (<intent-spec>) [::] <dummy-arg-name-list> <intent-spec> = IN | OUT | INOUT

generalization for pyf-files: INTENT (<intent-spec-list>) [::] <dummy-arg-name-list> <intent-spec> = IN | OUT | INOUT | CACHE | HIDE | OUT = <name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Entry**(*parent*, *item*)

ENTRY <entry-name> [([<dummy-arg-list>]) [<suffix>]] <suffix> = <proc-language-binding-spec> [RESULT (<result-name>)]

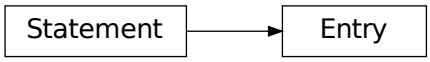
RESULT (<result-name>) [<proc-language-binding-spec>]

<proc-language-binding-spec> = <language-binding-spec> <language-binding-spec> =

BIND (C [, NAME = <scalar-char-initialization-expr>])

<dummy-arg> = <dummy-arg-name> | *

Inheritance



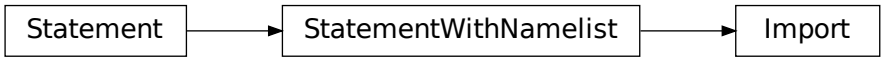
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Import**(*parent*, *item*)

IMPORT [[::] <import-name-list>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

fparser.one.block_statements.**ForallStmt**

alias of Forall

class fparser.one.block_statements.**SpecificBinding**(*parent*, *item*)

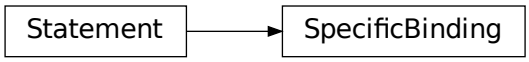
PROCEDURE [(<interface-name>)] [[, <binding-attr-list>] ::]
<binding-name> [=> <procedure-name>]

<binding-attr> = PASS [(<arg-name>)]

NOPASS
NON_OVERRIDABLE
DEFERRED
<access-spec>

<access-spec> = PUBLIC | PRIVATE

Inheritance



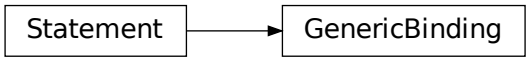
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**GenericBinding**(*parent*, *item*)

GENERIC [, <access-spec>] :: <generic-spec> => <binding-name-list>

Inheritance



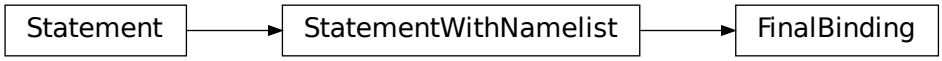
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**FinalBinding**(*parent*, *item*)

FINAL [::] <final-subroutine-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

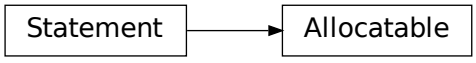
class fparser.one.block_statements.**Allocatable**(*parent*, *item*)

ALLOCATABLE [::] <object-name> [(<deferred-shape-spec-list>)]

[, <object-name>
[(<deferred-shape-spec-list>)]

]…

Inheritance



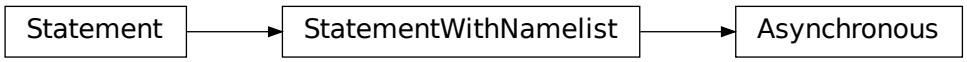
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Asynchronous**(*parent*, *item*)

ASYNCHRONOUS [::] <object-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

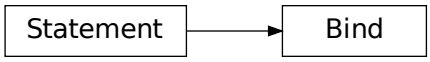
class fparser.one.block_statements.**Bind**(*parent*, *item*)

<language-binding-spec> [::] <bind-entity-list> <language-binding-spec> =

BIND (C [, NAME = <scalar-char-initialization-expr>])

<bind-entity> = <entity-name> | / <common-block-name> /

Inheritance



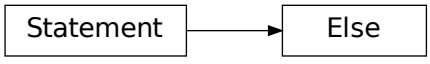
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Else**(*parent*, *item*)

ELSE [<if-construct-name>]

Inheritance



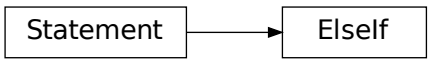
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**ElseIf**(*parent*, *item*)

ELSE IF (<scalar-logical-expr>) THEN [<if-construct-name>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Case**(*parent*, *item*)

CASE <case-selector> [<case-construct-name>] <case-selector> = (<case-value-range-list>) | DEFAULT <case-value-range> = <case-value>

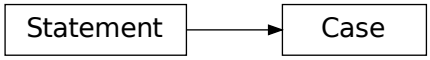
<case-value> :

: <case-value>

<case-value> : <case-value>

<case-value> = <scalar-(int|char|logical)-initialization-expr>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this item by parsing the associated line of code

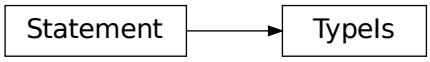
tofortran(*isfix=None*)

Return the Fortran for this object as a string

class fparser.one.block_statements.**TypeIs**(*parent*, *item*)

TYPE IS <type-selector> [<case-construct-name>] <type-selector> = (<type-value-range-list>) <type-value-range> = <case-value> <case-value> = <char>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this object by parsing the associated line of code

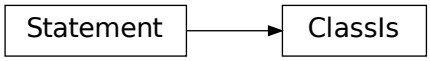
tofortran(*isfix=None*)

Create the Fortran representation of this object and return it as a string

class fparser.one.block_statements.**ClassIs**(*parent*, *item*)

CLASS <class-selector> <class-selector> = (IS <type-value-range-list> | DEFAULT) <type-value-range> = <case-value> <case-value> = <char>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this object by parsing the string

tofortran(*isfix=None*)

Returns the Fortran for this object as a string

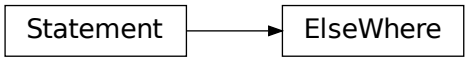
fparser.one.block_statements.**WhereStmt**

alias of Where

class fparser.one.block_statements.**ElseWhere**(*parent*, *item*)

ELSE WHERE (<mask-expr>) [<where-construct-name>] ELSE WHERE [<where-construct-name>]

Inheritance



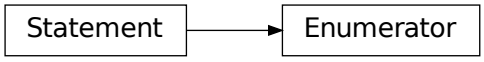
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Enumerator**(*parent*, *item*)

ENUMERATOR [::] <enumerator-list> <enumerator> = <named-constant> [= <scalar-int-initialization-expr>]

Inheritance



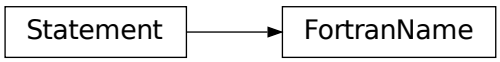
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**FortranName**(*parent*, *item*)

FORTRANNAME <name>

Inheritance



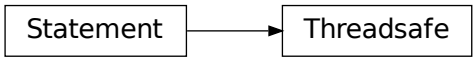
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Threadsafe**(*parent*, *item*)

THREADSAFE

Inheritance



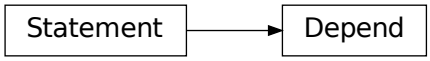
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Depend**(*parent*, *item*)

DEPEND (<name-list>) [::] <dummy-arg-name-list>

Inheritance



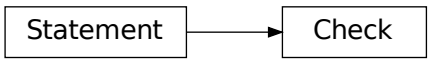
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Check**(*parent*, *item*)

CHECK (<c-int-scalar-expr>) [::] <name>

Inheritance



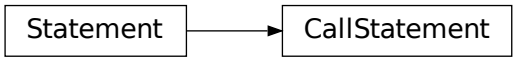
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**CallStatement**(*parent*, *item*)

CALLSTATEMENT <c-expr>

Inheritance



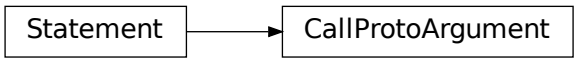
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**CallProtoArgument**(*parent*, *item*)

CALLPROTOARGUMENT <c-type-spec-list>

Inheritance



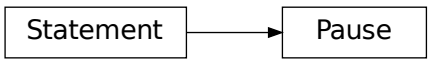
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Pause**(*parent*, *item*)

PAUSE [<char-literal-constant|int-literal-constant>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Comment**(*parent*, *item*)

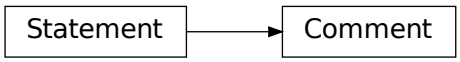
Attributes

content

[str] Content of the comment.

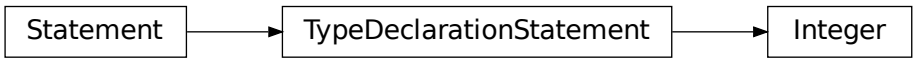
is_blank
[bool] When True then Comment represents blank line.

Inheritance



class fparser.one.block_statements.**Integer**(*parent*, *item*)

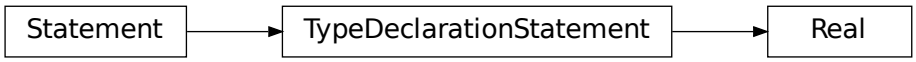
Inheritance



match(*pos=0*, *endpos=9223372036854775807*)
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Real**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)
Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**DoublePrecision**(*parent*, *item*)

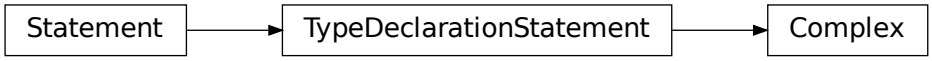
Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Complex**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**DoubleComplex**(*parent*, *item*)

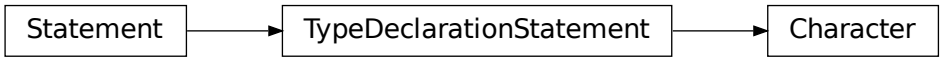
Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Character**(*parent*, *item*)

Inheritance

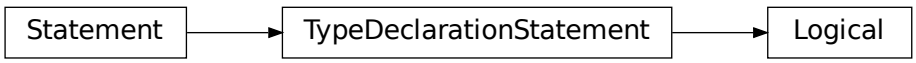


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Logical**(*parent*, *item*)

Inheritance

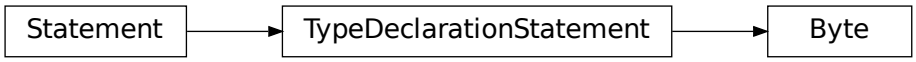


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Byte**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

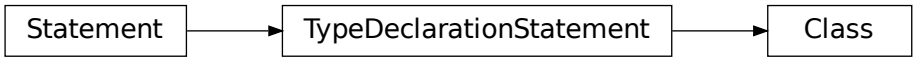
Matches zero or more characters at the beginning of the string.

fparser.one.block_statements.**TypeStmt**

alias of Type

class fparser.one.block_statements.**Class**(*parent*, *item*)

Inheritance



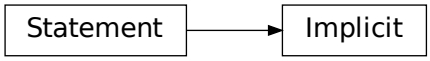
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.block_statements.**Implicit**(*parent*, *item*)

IMPLICIT <implicit-spec-list> IMPLICIT NONE <implicit-spec> = <declaration-type-spec> (<letter-spec-list>) <letter-spec> = <letter> [- <letter>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

Variables

- *intrinsic_type_spec*
- *declaration_type_spec*

fparser.one.block_statements.**intrinsic_type_spec**

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

```
[<class 'fparser.one.block_statements.SubprogramPrefix'>,
 <class 'fparser.one.typedecl_statements.Integer'>,
 <class 'fparser.one.typedecl_statements.Real'>,
 <class 'fparser.one.typedecl_statements.DoublePrecision'>,
 <class 'fparser.one.typedecl_statements.Complex'>,
 <class 'fparser.one.typedecl_statements.DoubleComplex'>,
 <class 'fparser.one.typedecl_statements.Character'>,
 <class 'fparser.one.typedecl_statements.Logical'>,
 <class 'fparser.one.typedecl_statements.Byte'>]
```

fparser.one.block_statements.**declaration_type_spec**

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

```
[<class 'fparser.one.block_statements.SubprogramPrefix'>,
 <class 'fparser.one.typedecl_statements.Integer'>,
 <class 'fparser.one.typedecl_statements.Real'>,
 <class 'fparser.one.typedecl_statements.DoublePrecision'>,
 <class 'fparser.one.typedecl_statements.Complex'>,
 <class 'fparser.one.typedecl_statements.DoubleComplex'>,
 <class 'fparser.one.typedecl_statements.Character'>,
 <class 'fparser.one.typedecl_statements.Logical'>,
 <class 'fparser.one.typedecl_statements.Byte'>,
 <class 'fparser.one.typedecl_statements.Type'>,
 <class 'fparser.one.typedecl_statements.Class'>]
```

fparser.one.parsefortran

Provides FortranParser.

Classes

- *FortranParser*: Parser of FortranReader structure.

class fparser.one.parsefortran.**FortranParser**(*reader*, *ignore_comments=True*)

Parser of FortranReader structure.

Use .parse() method for parsing, parsing result is saved in .block attribute.

Inheritance

analyze()

Attempts to analyse the parsed Fortran. It is not clear what for.

get_item()

Retrieves the next item from the reader.

parse()

Parses the program specified in the reader object.

put_item(*item*)

Pushes the given item to the reader.

fparser.one.statements

Fortran single line statements.

Classes

- *GeneralAssignment*: <variable> = <expr>
- *Assignment*: <variable> = <expr>
- *PointerAssignment*: <variable> = <expr>
- *Assign*: ASSIGN <label> TO <int-variable-name>
- *Call*: Call statement class
- *Goto*: GO TO <label>
- *ComputedGoto*: GO TO (<label-list>) [,] <scalar-int-expr>
- *AssignedGoto*: GO TO <int-variable-name> [(<label> [, <label>]...)]
- *Continue*: CONTINUE
- *Return*: RETURN [<scalar-int-expr>]
- *Stop*: STOP [<stop-code>]
- *Print*: PRINT <format> [, <output-item-list>]
- *Read*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Read0*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Read1*: Read0: READ (<io-control-spec-list>) [<input-item-list>]
- *Write*: WRITE (io-control-spec-list) [<output-item-list>]
- *Flush*: FLUSH <file-unit-number>
- *Wait*: WAIT (<wait-spec-list>)
- *Contains*: CONTAINS
- *Allocate*: ALLOCATE ([<type-spec> ::] <allocation-list> [, <alloc-opt-list>])
- *Deallocate*: DEALLOCATE (<allocate-object-list> [, <dealloc-opt-list>])
- *ModuleProcedure*: [MODULE] PROCEDURE [::] <procedure-name-list>
- *Access*: <access-spec> [[::] <access-id-list>]
- *Public*: <access-spec> [[::] <access-id-list>]
- *Private*: <access-spec> [[::] <access-id-list>]

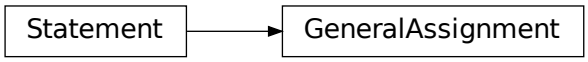
- *Close*: CLOSE (<close-spec-list>)
- *Cycle*: CYCLE [<do-construct-name>]
- *Backspace*: REWIND <file-unit-number>
- *Endfile*: REWIND <file-unit-number>
- *Rewind*: REWIND <file-unit-number>
- *Open*: OPEN (<connect-spec-list>)
- *Format*: FORMAT <format-specification>
- *Save*: SAVE [[::] <saved-entity-list>]
- *Data*: DATA <data-stmt-set> [[,] <data-stmt-set>]...
- *Nullify*: NULLIFY (<pointer-object-list>)
- *Use*: Parses USE statement.
- *Exit*: EXIT [<do-construct-name>]
- *Parameter*: PARAMETER (<named-constant-def-list>)
- *Equivalence*: EQUIVALENCE <equivalence-set-list>
- *Dimension*: DIMENSION [::] <array-name> (<array-spec>)
- *Target*: TARGET [::] <object-name> (<array-spec>)
- *Pointer*: POINTER [::] <pointer-decl-list>
- *Protected*: PROTECTED [::] <entity-name-list>
- *Volatile*: VOLATILE [::] <object-name-list>
- *Value*: VALUE [::] <dummy-arg-name-list>
- *ArithmeticIf*: IF (<scalar-numeric-expr>) <label> , <label> , <label>
- *Intrinsic*: INTRINSIC [::] <intrinsic-procedure-name-list>
- *Inquire*: INQUIRE (<inquire-spec-list>)
- *Sequence*: SEQUENCE
- *External*: EXTERNAL [::] <external-name-list>
- *Namelist*: NAMELIST / <namelist-group-name> / <namelist-group-object-list> [[,]
- *Common*: COMMON [/ [<common-block-name>] /] <common-block-object-list> [[,] / [<common-block-name>] / <common-block-object-list>]...
- *Optional*: OPTIONAL [::] <dummy-arg-name-list>
- *Intent*: INTENT (<intent-spec>) [::] <dummy-arg-name-list>
- *Entry*: ENTRY <entry-name> [([<dummy-arg-list>]) [<suffix>]]
- *Import*: IMPORT [[::] <import-name-list>]
- *ForallStmt*: FORALL <forall-header> <forall-assignment-stmt>
- *SpecificBinding*: PROCEDURE [(<interface-name>)] [[, <binding-attr-list>] ::]
- *GenericBinding*: GENERIC [, <access-spec>] :: <generic-spec> => <binding-name-list>
- *FinalBinding*: FINAL [::] <final-subroutine-name-list>

- *Allocatable*: ALLOCATABLE [::] <object-name> [(<deferred-shape-spec-list>)]
- *Asynchronous*: ASYNCHRONOUS [::] <object-name-list>
- *Bind*: <language-binding-spec> [::] <bind-entity-list>
- *Else*: ELSE [<if-construct-name>]
- *ElseIf*: ELSE IF (<scalar-logical-expr>) THEN [<if-construct-name>]
- *Case*: CASE <case-selector> [<case-construct-name>]
- *TypeIs*: TYPE IS <type-selector> [<case-construct-name>]
- *ClassIs*: CLASS <class-selector>
- *WhereStmt*: WHERE (<mask-expr>) <where-assignment-stmt>
- *ElseWhere*: ELSE WHERE (<mask-expr>) [<where-construct-name>]
- *Enumerator*: ENUMERATOR [::] <enumerator-list>
- *FortranName*: FORTRANNAME <name>
- *Threadsafe*: THREADSAFE
- *Depend*: DEPEND (<name-list>) [::] <dummy-arg-name-list>
- *Check*: CHECK (<c-int-scalar-expr>) [::] <name>
- *CallStatement*: CALLSTATEMENT <c-expr>
- *CallProtoArgument*: CALLPROTOARGUMENT <c-type-spec-list>
- *Pause*: PAUSE [<char-literal-constant|int-literal-constant>]
- *Comment*: Attributes

class fparser.one.statements.**GeneralAssignment**(*parent*, *item*)

<variable> = <expr> <pointer variable> => <expr>

Inheritance



item_re(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

match(*pos=0*, *endpos=9223372036854775807*)

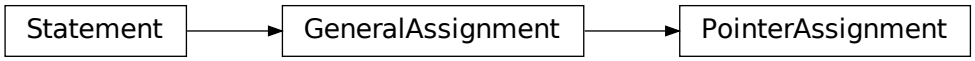
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Assignment**(*parent*, *item*)

Inheritance

class fparser.one.statements.**PointerAssignment**(*parent*, *item*)

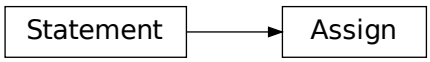
Inheritance



class fparser.one.statements.**Assign**(*parent*, *item*)

ASSIGN <label> TO <int-variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Call**(*parent*, *item*)

Call statement class CALL <procedure-designator> [([<actual-arg-spec-list>])]

<procedure-designator> = <procedure-name>

<proc-component-ref>
<data-ref> % <binding-name>

<actual-arg-spec> = [<keyword> =] <actual-arg> <actual-arg> = <expr>

<variable>

<procedure-name>
<proc-component-ref>
<alt-return-spec>

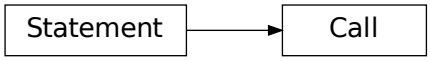
<alt-return-spec> = * <label>

<proc-component-ref> = <variable> % <procedure-component-name>

<variable> = <designator>

Call instance has attributes:
designator arg_list

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Parse the string containing the Call and store the designator and list of arguments (if any)

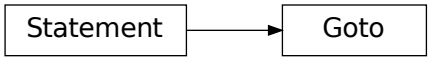
tofortran(*isfix=None*)

Returns the Fortran representation of this object as a string

class fparser.one.statements.**Goto**(*parent*, *item*)

GO TO <label>

Inheritance



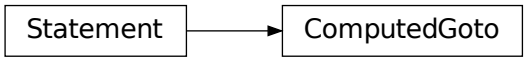
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**ComputedGoto**(*parent*, *item*)

GO TO (<label-list>) [,] <scalar-int-expr>

Inheritance



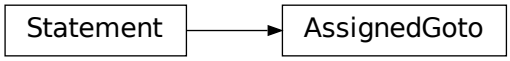
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**AssignedGoto**(*parent*, *item*)

GO TO <int-variable-name> [(<label> [, <label>]...)]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Continue**(*parent*, *item*)

CONTINUE

Inheritance

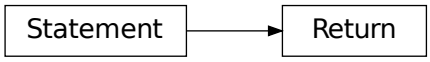
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Return**(*parent*, *item*)

RETURN [<scalar-int-expr>]

Inheritance



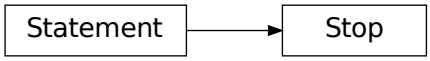
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Stop**(*parent*, *item*)

STOP [<stop-code>] <stop-code> = <scalar-char-constant> | <1-5-digit>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Print**(*parent*, *item*)

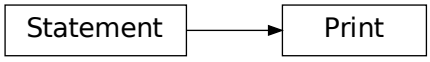
PRINT <format> [, <output-item-list>] <format> = <default-char-expr> | <label> | *

<output-item> = <expr> | <io-implied-do> <io-implied-do> = (<io-implied-do-object-list> , <implied-do-control>) <io-implied-do-object> = <input-item> | <output-item> <implied-do-control> = <do-variable>

= <scalar-int-expr> ,
<scalar-int-expr> [, <scalar-int-expr>]

<input-item> = <variable> | <io-implied-do>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Read**(*parent*, *item*)

Read0: READ (<io-control-spec-list>) [<input-item-list>]

<io-control-spec-list> = [UNIT =] <io-unit>

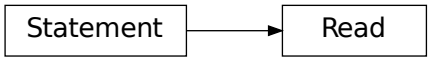
[FORMAT =] <format>
[NML =] <namelist-group-name>
ADVANCE = <scalar-default-char-expr>

...

Read1: READ <format> [, <input-item-list>]

<format> == <default-char-expr> | <label> | *

Inheritance

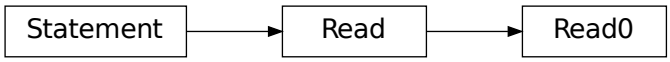


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

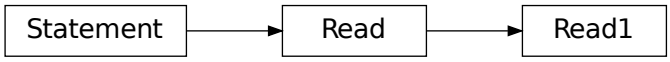
class fparser.one.statements.**Read0**(*parent*, *item*)

Inheritance



class fparser.one.statements.**Read1**(*parent*, *item*)

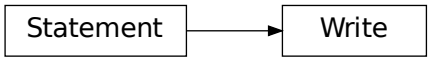
Inheritance



class fparser.one.statements.**Write**(*parent*, *item*)

WRITE (io-control-spec-list) [<output-item-list>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

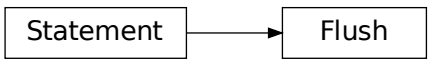
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Flush**(*parent*, *item*)

FLUSH <file-unit-number> FLUSH (<flush-spec-list>) <flush-spec> = [UNIT =] <file-unit-number>

IOSTAT = <scalar-int-variable>
IOMSG = <iomsg-variable>
ERR = <label>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

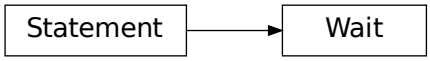
class fparser.one.statements.**Wait**(*parent*, *item*)

WAIT (<wait-spec-list>) <wait-spec> = [UNIT =] <file-unit-number>

END = <label>

EOR = <label>
ERR = <label>
ID = <scalar-int-expr>
IOMSG = <iomsg-variable>
IOSTAT = <scalar-int-variable>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Contains**(*parent*, *item*)

CONTAINS

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Allocate**(*parent*, *item*)

ALLOCATE ([<type-spec> ::] <allocation-list> [, <alloc-opt-list>]) <alloc-opt> = STAT = <stat-variable>

ERRMSG = <errmsg-variable>
SOURCE = <source-expr>

<allocation> = <allocate-object> [(<allocate-shape-spec-list>)]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Process the ALLOCATE statement and store the various entities being allocated in self.items. Any type-specification is stored in self.spec.

Raises
ParseError – if an invalid type-specification is used

tofortran(*isfix=None*)

Create the Fortran code for this ALLOCATE statement

Parameters
isfix (*bool*) – whether or not to generate fixed-format code

Returns
Fortran code

Return type
str

class fparser.one.statements.**Deallocate**(*parent*, *item*)

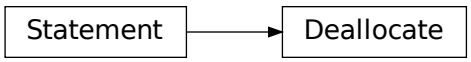
DEALLOCATE (<allocate-object-list> [, <dealloc-opt-list>]) <allocate-object> = <variable-name>

<structure-component>

<structure-component> = <data-ref> <dealloc-opt> = STAT = <stat-variable>

ERRMSG = <errmsg-variable>

Inheritance



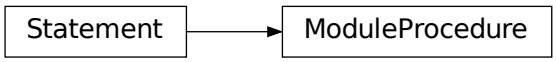
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**ModuleProcedure**(*parent*, *item*)

[MODULE] PROCEDURE [::] <procedure-name-list>

Inheritance



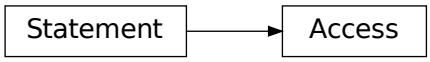
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Access**(*parent*, *item*)

<access-spec> [[::] <access-id-list>] <access-spec> = PUBLIC | PRIVATE <access-id> = <use-name> | <generic-spec>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Public**(*parent*, *item*)

Inheritance

class fparser.one.statements.**Private**(*parent*, *item*)

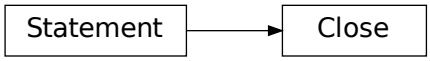
Inheritance

class fparser.one.statements.**Close**(*parent*, *item*)

CLOSE (<close-spec-list>) <close-spec> = [UNIT =] <file-unit-number>

IOSTAT = <scalar-int-variable>
IOMSG = <iomsg-variable>
ERR = <label>
STATUS = <scalar-default-char-expr>

Inheritance



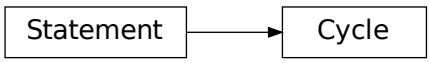
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Cycle**(*parent*, *item*)

CYCLE [<do-construct-name>]

Inheritance

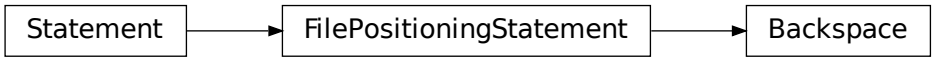


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

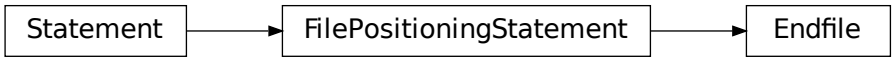
class fparser.one.statements.**Backspace**(*parent*, *item*)

Inheritance



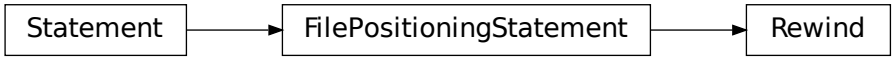
class fparser.one.statements.**Endfile**(*parent*, *item*)

Inheritance



class fparser.one.statements.**Rewind**(*parent*, *item*)

Inheritance



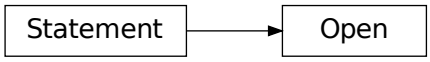
class fparser.one.statements.**Open**(*parent*, *item*)

OPEN (<connect-spec-list>) <connect-spec> = [UNIT =] <file-unit-number>

ACCESS = <scalar-default-char-expr>

..

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Format**(*parent*, *item*)

FORMAT <format-specification> <format-specification> = ([<format-item-list>]) <format-item> = [<r>] <data-edit-descr>

<control-edit-descr>
<char-string-edit-descr>
[<r>] (<format-item-list>)

<data-edit-descr> = I <w> [. <m>]

B <w> [. <m>]

...

<r|w|m|d|e> = <int-literal-constant> <v> = <signed-int-literal-constant> <control-edit-descr> = <position-edit-descr>

[<r>] /
:

...

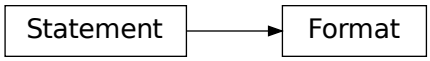
<position-edit-descr> = T <n>

TL <n>

...

<sign-edit-descr> = SS | SP | S ...

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

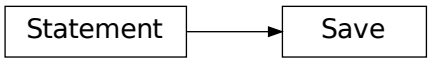
class fparser.one.statements.**Save**(*parent*, *item*)

SAVE [[::] <saved-entity-list>] <saved-entity> = <object-name>

<proc-pointer-name>
/ <common-block-name> /

<proc-pointer-name> = <name> <object-name> = <name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Data**(*parent*, *item*)

DATA <data-stmt-set> [[,] <data-stmt-set>]... <data-stmt-set> = <data-stmt-object-list> / <data-stmt-value-list> / <data-stmt-object> = <variable> | <data-implied-do> <data-implied-do> = (<data-i-do-object-list> ,

<data-i-do-variable> = <scalar-int-expr> , <scalar-int-expr> [, <scalar-int-expr>])

<data-i-do-object> = <array-element>

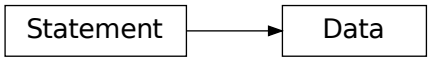
<scalar-structure-component>
<data-implied-do>

<data-i-do-variable> = <scalar-int-variable> <variable> = <designator> <designator> = <object-name>

<array-element>
<array-section>
<structure-component>
<substring>

<array-element> = <data-ref> <array-section> = <data-ref> [(<substring-range>)]

Inheritance



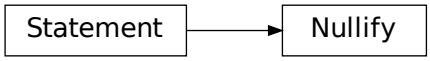
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Nullify**(*parent*, *item*)

NULLIFY (<pointer-object-list>) <pointer-object> = <variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Use**(*parent*, *item*)

Parses USE statement.

Parameters

Statement (*class*) – Base fparser class.

Raises

AnalyzeError – If entity name is not in module.

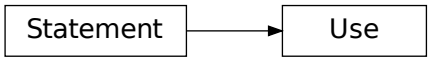
Fortran syntax construct:

USE [[, <module-nature>] ::] <module-name> [, <rename-list>] USE [[, <module-nature>] ::] <module-name> , ONLY : [<only-list>] <module-nature> = INTRINSIC | NON_INTRINSIC <rename> = <local-name> => <use-name>

OPERATOR (<local-defined-operator>) => OPERATOR (<use-defined-operator>)

<only> = <generic-spec> | <only-use-name> | <rename> <only-use-name> = <use-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

populate_use_provides(*all_mod_provides*, *use_provides*, *name*, *rename=None*)

Checks for entity name in the module

process_item()

Parse the string containing the Use and store the module name and list of attributes (if any)

tofortran(*isfix=None*)

Returns the Fortran representation of this object as a string

Parameters
isfix (*bool*) – Whether or not to generated fixed-format Fortran

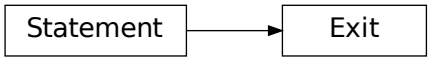
Returns
Fortran representation of this object

Return type
str

class fparser.one.statements.**Exit**(*parent*, *item*)

EXIT [<do-construct-name>]

Inheritance



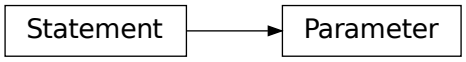
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Parameter**(*parent*, *item*)

PARAMETER (<named-constant-def-list>) <named-constant-def> = <named-constant> = <initialization-expr>

Inheritance



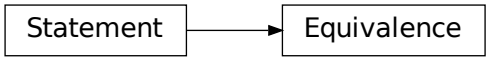
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Equivalence**(*parent*, *item*)

EQUIVALENCE <equivalence-set-list> <equivalence-set> = (<equivalence-object> , <equivalence-object-list>) <equivalence-object> = <variable-name> | <array-element> | <substring>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

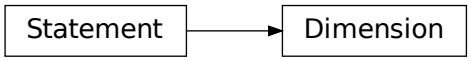
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Dimension**(*parent*, *item*)

DIMENSION [::] <array-name> (<array-spec>)

[, <array-name> (<array-spec>)]...

Inheritance



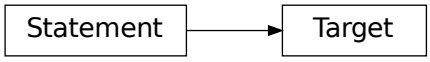
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Target**(*parent*, *item*)

TARGET [::] <object-name> (<array-spec>)
[, <object-name> (<array-spec>)]...

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

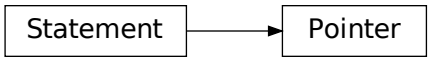
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Pointer**(*parent*, *item*)

POINTER [::] <pointer-decl-list> <pointer-decl> = <object-name> [(<deferred-shape-spec-list>)]

<proc-entity-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Protected**(*parent*, *item*)

PROTECTED [::] <entity-name-list>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Volatile**(*parent*, *item*)

VOLATILE [::] <object-name-list>

Inheritance

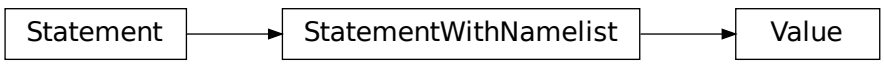
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Value**(*parent*, *item*)

VALUE [::] <dummy-arg-name-list>

Inheritance



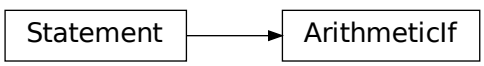
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**ArithmeticIf**(*parent*, *item*)

IF (<scalar-numeric-expr>) <label> , <label> , <label>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Intrinsic**(*parent*, *item*)

INTRINSIC [::] <intrinsic-procedure-name-list>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Inquire**(*parent*, *item*)

INQUIRE (<inquire-spec-list>) INQUIRE (IOLENGTH = <scalar-int-variable>) <output-item-list>

<inquire-spec> = [UNIT =] <file-unit-number>

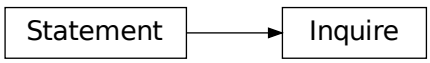
FILE = <file-name-expr>

...

<output-item> = <expr>

<io-implied-do>

Inheritance



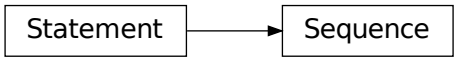
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Sequence**(*parent*, *item*)

SEQUENCE

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**External**(*parent*, *item*)

EXTERNAL [::] <external-name-list>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

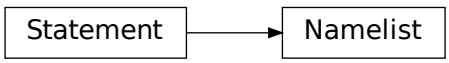
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Namelist**(*parent*, *item*)

NAMELIST / <namelist-group-name> / <namelist-group-object-list> [[,]
/ <namelist-group-name> / <namelist-group-object-list>]...

<namelist-group-object> = <variable-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

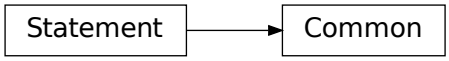
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Common**(*parent*, *item*)

COMMON [/ [<common-block-name>] /] <common-block-object-list> [[,] / [<common-block-name>] / <common-block-object-list>]... <common-block-object> = <variable-name> [(<explicit-shape-spec-list>)]

<proc-pointer-name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Optional**(*parent*, *item*)

OPTIONAL [::] <dummy-arg-name-list> <dummy-arg-name> = <name>

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

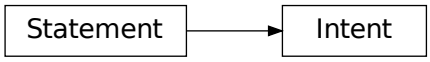
Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Intent**(*parent*, *item*)

INTENT (<intent-spec>) [::] <dummy-arg-name-list> <intent-spec> = IN | OUT | INOUT

generalization for pyf-files: INTENT (<intent-spec-list>) [::] <dummy-arg-name-list> <intent-spec> = IN | OUT | INOUT | CACHE | HIDE | OUT = <name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Entry**(*parent*, *item*)

ENTRY <entry-name> [([<dummy-arg-list>]) [<suffix>]] <suffix> = <proc-language-binding-spec> [RESULT (<result-name>)]

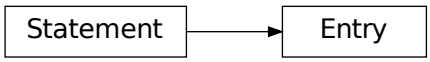
RESULT (<result-name>) [<proc-language-binding-spec>]

<proc-language-binding-spec> = <language-binding-spec> <language-binding-spec> =

BIND (C [, NAME = <scalar-char-initialization-expr>])

<dummy-arg> = <dummy-arg-name> | *

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Import**(*parent*, *item*)

IMPORT [[::] <import-name-list>]

Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

fparser.one.statements.**ForallStmt**

alias of Forall

class fparser.one.statements.**SpecificBinding**(*parent*, *item*)

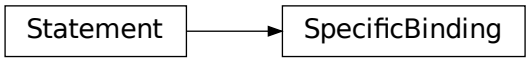
PROCEDURE [(<interface-name>)] [[, <binding-attr-list>] ::]
<binding-name> [=> <procedure-name>]

<binding-attr> = PASS [(<arg-name>)]

NOPASS
NON_OVERRIDABLE
DEFERRED
<access-spec>

<access-spec> = PUBLIC | PRIVATE

Inheritance



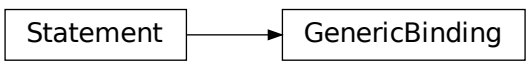
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**GenericBinding**(*parent*, *item*)

GENERIC [, <access-spec>] :: <generic-spec> => <binding-name-list>

Inheritance



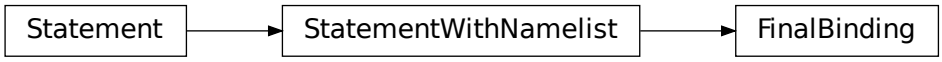
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**FinalBinding**(*parent*, *item*)

FINAL [::] <final-subroutine-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

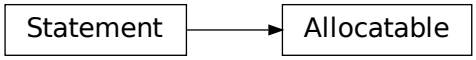
class fparser.one.statements.**Allocatable**(*parent*, *item*)

ALLOCATABLE [::] <object-name> [(<deferred-shape-spec-list>)]

[, <object-name>
[(<deferred-shape-spec-list>)]

]…

Inheritance



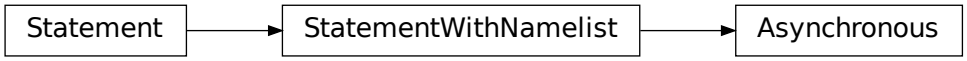
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Asynchronous**(*parent*, *item*)

ASYNCHRONOUS [::] <object-name-list>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

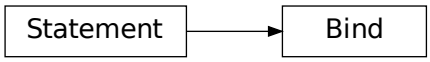
class fparser.one.statements.**Bind**(*parent*, *item*)

<language-binding-spec> [::] <bind-entity-list> <language-binding-spec> =

BIND (C [, NAME = <scalar-char-initialization-expr>])

<bind-entity> = <entity-name> | / <common-block-name> /

Inheritance



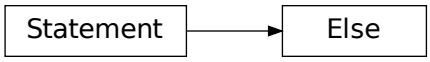
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Else**(*parent*, *item*)

ELSE [<if-construct-name>]

Inheritance



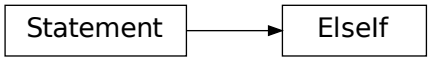
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**ElseIf**(*parent*, *item*)

ELSE IF (<scalar-logical-expr>) THEN [<if-construct-name>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Case**(*parent*, *item*)

CASE <case-selector> [<case-construct-name>] <case-selector> = (<case-value-range-list>) | DEFAULT <case-value-range> = <case-value>

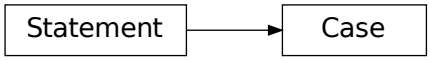
<case-value> :

: <case-value>

<case-value> : <case-value>

<case-value> = <scalar-(int|char|logical)-initialization-expr>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this item by parsing the associated line of code

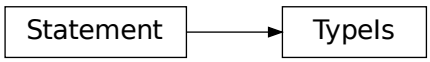
tofortran(*isfix=None*)

Return the Fortran for this object as a string

class fparser.one.statements.**TypeIs**(*parent*, *item*)

TYPE IS <type-selector> [<case-construct-name>] <type-selector> = (<type-value-range-list>) <type-value-range> = <case-value> <case-value> = <char>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this object by parsing the associated line of code

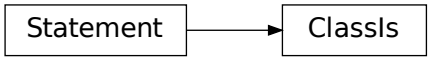
tofortran(*isfix=None*)

Create the Fortran representation of this object and return it as a string

class fparser.one.statements.**ClassIs**(*parent*, *item*)

CLASS <class-selector> <class-selector> = (IS <type-value-range-list> | DEFAULT) <type-value-range> = <case-value> <case-value> = <char>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

process_item()

Populate the state of this object by parsing the string

tofortran(*isfix=None*)

Returns the Fortran for this object as a string

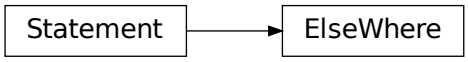
fparser.one.statements.**WhereStmt**

alias of Where

class fparser.one.statements.**ElseWhere**(*parent*, *item*)

ELSE WHERE (<mask-expr>) [<where-construct-name>] ELSE WHERE [<where-construct-name>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Enumerator**(*parent*, *item*)

ENUMERATOR [::] <enumerator-list> <enumerator> = <named-constant> [= <scalar-int-initialization-expr>]

Inheritance

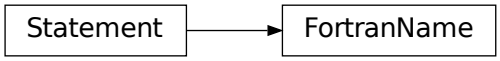
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**FortranName**(*parent*, *item*)

FORTRANNAME <name>

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Threadsafe**(*parent*, *item*)

THREADSAFE

Inheritance

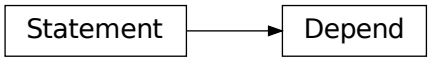
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Depend**(*parent*, *item*)

DEPEND (<name-list>) [::] <dummy-arg-name-list>

Inheritance



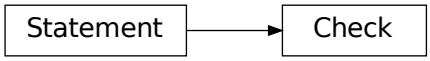
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Check**(*parent*, *item*)

CHECK (<c-int-scalar-expr>) [::] <name>

Inheritance



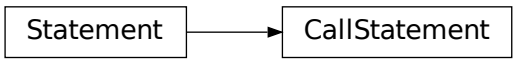
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**CallStatement**(*parent*, *item*)

CALLSTATEMENT <c-expr>

Inheritance



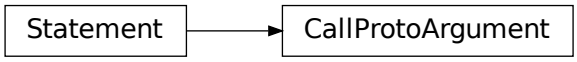
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**CallProtoArgument**(*parent*, *item*)

CALLPROTOARGUMENT <c-type-spec-list>

Inheritance



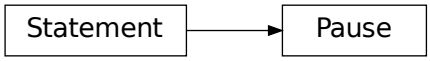
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Pause**(*parent*, *item*)

PAUSE [<char-literal-constant|int-literal-constant>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.statements.**Comment**(*parent*, *item*)

Attributes

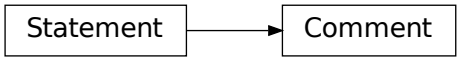
content

[str] Content of the comment.

is_blank

[bool] When True then Comment represents blank line.

Inheritance



fparser.one.tests

Submodules

fparser.one.tests.test_scripts

Test fparser one scripts

fparser.one.typedecl_statements

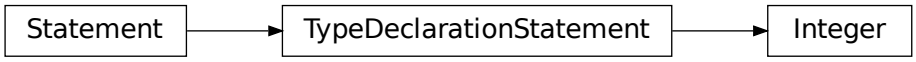
Fortran type-declaration statements.

Classes

- *Integer*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Real*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *DoublePrecision*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Complex*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *DoubleComplex*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Character*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Logical*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Byte*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *TypeStmt*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Class*: <declaration-type-spec> [[, <attr-spec>] ::] <entity-decl-list>
- *Implicit*: IMPLICIT <implicit-spec-list>

class fparser.one.typedecl_statements.**Integer**(*parent*, *item*)

Inheritance

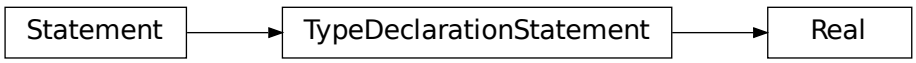


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Real**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**DoublePrecision**(*parent*, *item*)

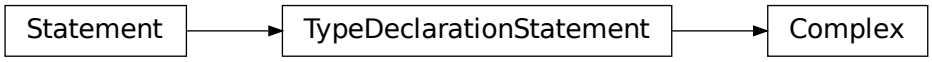
Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Complex**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**DoubleComplex**(*parent*, *item*)

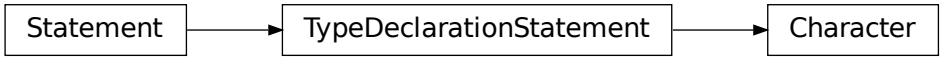
Inheritance

match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Character**(*parent*, *item*)

Inheritance

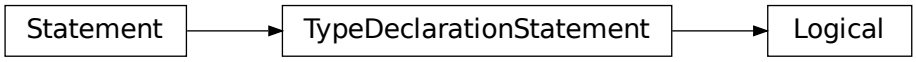


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Logical**(*parent*, *item*)

Inheritance

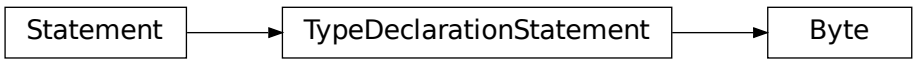


match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Byte**(*parent*, *item*)

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

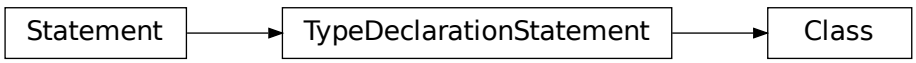
Matches zero or more characters at the beginning of the string.

fparser.one.typedecl_statements.**TypeStmt**

alias of Type

class fparser.one.typedecl_statements.**Class**(*parent*, *item*)

Inheritance



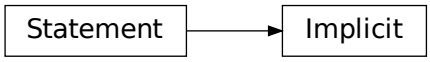
match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

class fparser.one.typedecl_statements.**Implicit**(*parent*, *item*)

IMPLICIT <implicit-spec-list> IMPLICIT NONE <implicit-spec> = <declaration-type-spec> (<letter-spec-list>) <letter-spec> = <letter> [- <letter>]

Inheritance



match(*pos=0*, *endpos=9223372036854775807*)

Matches zero or more characters at the beginning of the string.

Variables

- *intrinsic_type_spec*
- *declaration_type_spec*

fparser.one.typedecl_statements.**intrinsic_type_spec**

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

```
[<class 'fparser.one.typedecl_statements.Integer'>,
 <class 'fparser.one.typedecl_statements.Real'>,
 <class 'fparser.one.typedecl_statements.DoublePrecision'>,
 <class 'fparser.one.typedecl_statements.Complex'>,
 <class 'fparser.one.typedecl_statements.DoubleComplex'>,
 <class 'fparser.one.typedecl_statements.Character'>,
 <class 'fparser.one.typedecl_statements.Logical'>,
 <class 'fparser.one.typedecl_statements.Byte'>]
```

fparser.one.typedecl_statements.**declaration_type_spec**

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list. The argument must be an iterable if specified.

```
[<class 'fparser.one.typedecl_statements.Integer'>,
 <class 'fparser.one.typedecl_statements.Real'>,
 <class 'fparser.one.typedecl_statements.DoublePrecision'>,
 <class 'fparser.one.typedecl_statements.Complex'>,
 <class 'fparser.one.typedecl_statements.DoubleComplex'>,
 <class 'fparser.one.typedecl_statements.Character'>,
 <class 'fparser.one.typedecl_statements.Logical'>,
 <class 'fparser.one.typedecl_statements.Byte'>,
 <class 'fparser.one.typedecl_statements.Type'>,
 <class 'fparser.one.typedecl_statements.Class'>]
```

fparser.scripts

Submodules

fparser.scripts.fparser2

Example script to parse a Fortran program using fparser

fparser.scripts.fparser2_bench

Generates a large Fortran program in memory and then measures how long it takes fparser2 to parse it. This is based on the benchmark suggested by Ondřej Čertík via Ioannis Nikiteas.

fparser.scripts.parse

fparser.scripts.read

Python script with command line options which calls the Fortran File Reader with the supplied filename(s) and outputs the reader's representation of the code(s).

fparser.scripts.script_options

Functions

- *set_read_options()*: Undocumented.
- *set_parse_options()*: Undocumented.
- *get_fortran_code_group()*: Undocumented.

fparser.scripts.script_options.**set_read_options**(*parser*)

fparser.scripts.script_options.**set_parse_options**(*parser*)

fparser.scripts.script_options.**get_fortran_code_group**(*parser*)

fparser.two

Submodules

fparser.two.C99Preprocessor

C99 Preprocessor Syntax Rules.

fparser.two.Fortran2003

Fortran 2003 Syntax Rules.

fparser.two.Fortran2008

The file implements the Fortran2008 rules as defined in https://j3-fortran.org/doc/year/10/10-007r1.pdf

fparser.two.parser

This file provides utilities to create a Fortran parser suitable for a particular standard.

fparser.two.pattern_tools

Tools for constructing patterns.

Permission to use, modify, and distribute this software is given under the terms of the NumPy License. See http://scipy.org.

NO WARRANTY IS EXPRESSED OR IMPLIED. USE AT YOUR OWN RISK. Author: Pearu Peterson <pearu@cens.ioc.ee> Created: Oct 2006

fparser.two.symbol_table

The fparser2 symbol-table module. Defines various classes as well as the single, global SYMBOL_TABLES instance. The latter is a container for all of the top-level scoping units encountered during parsing.

Classes

- *SymbolTables*: Class encapsulating functionality for the global symbol-tables object.
- *SymbolTable*: Class implementing a single symbol table.

class fparser.two.symbol_table.**SymbolTables**

Class encapsulating functionality for the global symbol-tables object. This is a container for all symbol tables constructed while parsing code. All names are converted to lower case (since Fortran is not case sensitive).

Inheritance

add(*name*)

Add a new symbol table with the supplied name. The name will be converted to lower case if necessary.

Parameters
name (*str*) – the name for the new table.

Returns
the new symbol table.

Return type
fparser.two.symbol_table.SymbolTable

Raises
SymbolTableError – if there is already an entry with the supplied name.

clear()

Deletes any stored SymbolTables but retains the stored list of classes that define scoping units.

property current_scope

Returns
the symbol table for the current scoping unit or None.

Return type
fparser.two.symbol_table.SymbolTable or NoneType

enable_checks(*value*)

Sets whether or not to enable consistency checks in every symbol table that is created during a parse.

Parameters
value (*bool*) – whether or not checks are enabled.

enter_scope(*name*)

Called when the parser enters a new scoping region (i.e. when it encounters one of the classes listed in *_scoping_unit_classes*). Sets the 'current scope' to be the symbol table with the supplied name. If we are not currently within a tree of scoping regions then a new entry is created in the internal dict of symbol tables. If there is an existing tree then a new table is created and added to the bottom.

Parameters
name (*str*) – name of the scoping region.

`exit_scope()`

Marks the end of the processing of the current scoping unit. Since we are exiting the current scoping region, the new 'current scoping region' will be its parent.

Raises
SymbolTableError – if there is no current scope from which to exit.

`lookup(name)`

Find the named symbol table and return it.

Parameters
name (*str*) – the name of the required symbol table (not case sensitive).

Returns
the named symbol table.

Return type
fparser.two.symbol_table.SymbolTable

`remove(name)`

Removes the named symbol table and any descendants it may have. When searching for the named table, the current scope takes priority followed by the list of top-level symbol tables.

Parameters
name (*str*) – the name of the symbol table to remove (not case sensitive).

Raises
SymbolTableError – if the named symbol table is not in the current scope or in the list of top-level symbol tables.

`property scoping_unit_classes`

Returns
the fparser2 classes that are taken to mark the start of a new scoping region.

Return type
list of types

`class fparser.two.symbol_table.SymbolTable(name, parent=None, checking_enabled=False)`

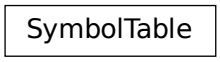
Class implementing a single symbol table.

Since this functionality is not yet fully mature, checks that new symbols don't clash with existing symbols are disabled by default. Once #201 is complete it is planned to switch this so that the checks are instead enabled by default.

Parameters

- **name** (*str*) – the name of this scope. Will be the name of the associated module or routine.
- **parent** (*fparser.two.symbol_table.SymbolTable.Symbol*) – the symbol table within which this one is nested (if any).
- **checking_enabled** (*bool*) – whether or not validity checks are performed for symbols added to the table.

Inheritance



class Symbol(*name*, *primitive_type*)

property name

Alias for field number 0

property primitive_type

Alias for field number 1

add_child(*child*)

Adds a child symbol table (scoping region nested within this one).

Parameters

child (*fparser.two.symbol_table.SymbolTable*) – the nested symbol table.

Raises

TypeError – if the supplied child is not a SymbolTable.

add_data_symbol(*name*, *primitive_type*)

Creates a new Symbol with the specified properties and adds it to the symbol table. The supplied name is converted to lower case.

TODO #201 add support for other symbol properties (kind, shape and visibility).

Parameters

- **name** (*str*) – the name of the symbol.
- **primitive_type** (*str*) – the primitive type of the symbol.

Raises

- **TypeError** – if any of the supplied parameters are of the wrong type.
- ***SymbolTableError*** – if the symbol table already contains an entry with the supplied name.

add_use_symbols(*name*, *only_list=None*, *rename_list=None*)

Creates an entry in the table for the USE of a module with the supplied name. If no *only_list* is supplied then this USE represents a wildcard import of all public symbols in the named module. If the USE statement has an ONLY clause but without any named symbols then *only_list* should be an empty list.

A USE can also have one or more rename entries *without* an only list.

Parameters

- **name** (*str*) – the name of the module being imported via a USE. Not case sensitive.
- **only_list** (*Optional[List[Tuple[str, str | NoneType]]]*) – if there is an 'only:' clause on the USE statement then this contains a list of tuples, each holding the local name of the symbol and its name in the module from which it is imported. These names are case insensitive.

- **rename_list** (Optional[List[Tuple[str, str]]]) – a list of symbols that are renamed from the scope being imported. Each entry is a tuple containing the name in the local scope and the corresponding name in the module from which it is imported. These names are case insensitive.

property children

Returns
the child (nested) symbol tables, if any.

Return type
list of *fparser.two.symbol_table.SymbolTable*

del_child(*name*)

Removes the named symbol table.

Parameters
name (*str*) – the name of the child symbol table to delete (not case sensitive).

Raises
KeyError – if the named table is not a child of this one.

lookup(*name*)

Lookup the symbol with the supplied name.

Parameters
name (*str*) – the name of the symbol to lookup (not case sensitive).

Returns
the named symbol.

Return type
fparser.two.symbol_table.SymbolTable.Symbol

Raises
KeyError – if the named symbol cannot be found in this or any parent scope.

property name

Returns
the name of this symbol table (scoping region).

Return type
str

property parent

Returns
the parent symbol table (scoping region) that contains this one (if any).

Return type
fparser.two.symbol_table.SymbolTable or NoneType

property root

Returns
the top-level symbol table that contains the current scoping region (symbol table).

Return type
fparser.two.symbol_table.SymbolTable

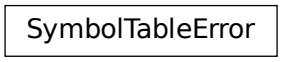
Exceptions

- *SymbolTableError*: Base class exception for symbol-table related errors.

exception fparser.two.symbol_table.**SymbolTableError**

Base class exception for symbol-table related errors.

Inheritance



Variables

- *SYMBOL_TABLES*

fparser.two.symbol_table.**SYMBOL_TABLES**

The single, global container for all symbol tables constructed while parsing.

```
<fparser.two.symbol_table.SymbolTables object at 0x7ff8c170b8d0>
```

fparser.two.utils

Base classes and exception handling for Fortran parser.

The Sphinx/AutoAPI documentation is *here.* or you can use the navigation pane on the left.

CHAPTER

TWO

DOXYGEN-GENERATED DOCUMENTATION

PYTHON MODULE INDEX

f

INDEX

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W