Top-level components

The Python interpreter can get its input from a number of sources: from a script passed to it as standard input or as program argument, typed in interactively, from a module source file, etc. This chapter gives the syntax used in these cases.

Complete Python programs

While a language specification need not prescribe how the language interpreter is invoked, it is useful to have a notion of a complete Python program. A complete Python program is executed in a minimally initialized environment: all built-in and standard modules are available, but none have been initialized, except for :mod:`sys` (various system services), :mod:`builtins` (built-in functions, exceptions and None) and :mod:`__main__`. The latter is used to provide the local and global namespace for execution of the complete program.

The syntax for a complete Python program is that for file input, described in the next section.

The interpreter may also be invoked in interactive mode; in this case, it does not read and execute a complete program but reads and executes one statement (possibly compound) at a time. The initial environment is identical to that of a complete program; each statement is executed in the namespace of :mod:`__main__`.

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 43);** *backlink*

Unknown interpreted text role "mod".

</div>

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 49)**

Unknown directive type "index".

```
.. index::
   single: UNIX
   single: Windows
   single: command line
   single: standard input
```

</div>

A complete program can be passed to the interpreter in three forms: with the :option:`-c` *string* command line option, as a file passed as the first command line argument, or as standard input. If the file or standard input is a tty device, the interpreter enters interactive mode; otherwise, it executes the file as a complete program.

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 55);** *backlink*

Unknown interpreted text role "option".

</div>

File input

All input read from non-interactive files has the same form:

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 69)**

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   file_input: (NEWLINE | `statement`)*
```

</div>

This syntax is used in the following situations:

- when parsing a complete Python program (from a file or from a string);
- when parsing a module;
- when parsing a string passed to the :func:`exec` function;

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 78);** *backlink*

Unknown interpreted text role "func".

</div>

Interactive input

Input in interactive mode is parsed using the following grammar:

<div style="border:1px solid">

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`**, line 88)**

Unknown directive type "productionlist".

```
.. productionlist:: python-grammar
   interactive_input: [`stmt_list`] NEWLINE | `compound_stmt` NEWLINE
```

</div>

Note that a (top-level) compound statement must be followed by a blank line in interactive mode; this is needed to help the parser detect the end of the input.

Expression input

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`, **line 100**)
>
> Unknown directive type "index".
>
> ```
> .. index:: single: input
> ```

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`, **line 101**)
>
> Unknown directive type "index".
>
> ```
> .. index:: builtin: eval
> ```

:func:`eval` is used for expression input. It ignores leading whitespace. The string argument to :func:`eval` must have the following form:

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`, **line 103**); *backlink*
>
> Unknown interpreted text role "func".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`, **line 103**); *backlink*
>
> Unknown interpreted text role "func".

> **System Message: ERROR/3** (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\reference\[cpython-main][Doc][reference]toplevel_components.rst`, **line 106**)
>
> Unknown directive type "productionlist".
>
> ```
> .. productionlist:: python-grammar
>    eval_input: `expression_list` NEWLINE*
> ```