



Curtin University

# Interactive Help

**ISYS5002, School of Marketing and Management**

***ELECTRONIC WARNING NOTICE FOR COPYRIGHT  
STATUTORY LICENCES***

**WARNING**

This material has been reproduced and communicated to you by or on behalf of **Curtin University** in accordance with section 113P of the *Copyright Act 1968 (the Act)*

The material in this communication may be subject to copyright under the Act. Any further reproduction or communication of this material by you may be the subject of copyright protection under the Act.

Do not remove this notice.



I acknowledge the traditional custodians of  
the land on which I work and live, and  
recognise their continuing connection to land,  
water and community. I pay respect to elders  
past, present and emerging.





...

Welcome to Python 3.7's help utility!

If this is your first time using Python, you should definitely check out the tutorial on the Internet at <https://docs.python.org/3.7/tutorial/>.

Enter the name of any module, keyword, or topic to get help on writing Python programs and using Python modules. To quit this help utility and return to the interpreter, just type "quit".

To get a list of available modules, keywords, symbols, or topics, type "modules", "keywords", "symbols", or "topics". Each module also comes with a one-line summary of what it does; to list the modules whose name or summary contain a given string such as "spam", type "modules spam".

help>



## `help(print)`

 Help on built-in function print in module builtins:

```
print(...)  
    print(value, ..., sep=' ', end='\n', file=sys.stdout, flush=False)
```

Prints the values to a stream, or to sys.stdout by default.

Optional keyword arguments:

file: a file-like object (stream); defaults to the current sys.stdout.

sep: string inserted between values, default a space.

end: string appended after the last value, default a newline.

flush: whether to forcibly flush the stream.

```
▶ import io
  dir(io)
```

```
↳ ['BlockingIOError',
   'BufferedIOBase',
   'BufferedRWPair',
   'BufferedRandom',
   'BufferedReader',
   'BufferedWriter',
   'BytesIO',
   'DEFAULT_BUFFER_SIZE',
   'FileIO',
   'IOBase',
   'IncrementalNewlineDecoder',
   'OpenWrapper',
   'RawIOBase',
   'SEEK_CUR',
   'SEEK_END',
   'SEEK_SET',
   'StringIO',
```

## `help(dir)`

☞ Help on built-in function dir in module builtins:

```
dir(...)
dir([object]) -> list of strings
```

If called without an argument, return the names in the current scope.

Else, return an alphabetized list of names comprising (some of) the attributes of the given object, and of attributes reachable from it.

If the object supplies a method named `__dir__`, it will be used; otherwise the default `dir()` logic is used and returns:


- for a module object: the module's attributes.

- for a class object: its attributes, and recursively the attributes of its bases.

- for any other object: its attributes, its class's attributes, and recursively the attributes of its class's base classes.



## `help(io.open)`

 Help on built-in function open in module io:

```
open(file, mode='r', buffering=-1, encoding=None, errors=None, newline=None, closefd=True, opener=None)
    Open file and return a stream.  Raise OSError upon failure.
```

`file` is either a text or byte string giving the name (and the path if the file isn't in the current working directory) of the file to be opened or an integer file descriptor of the file to be wrapped. (If a file descriptor is given, it is closed when the returned I/O object is closed, unless `closefd` is set to `False`.)

`mode` is an optional string that specifies the mode in which the file is opened. It defaults to `'r'` which means open for reading in text mode. Other common values are `'w'` for writing (truncating the file if it already exists), `'x'` for creating and writing to a new file, and `'a'` for appending (which on some Unix systems, means that all writes append to the end of the file regardless of the current seek position). In text mode, if encoding is not specified the encoding used is platform dependent: `locale.getpreferredencoding(False)` is called to get the current locale encoding. (For reading and writing raw bytes use binary mode and leave encoding unspecified.) The available modes are:





```
def addTwo(num):  
    '''This function will add 2 to the input'''  
    return num + 2
```



```
help(addTwo)|
```

```
Help on function addTwo in module __main__:
```

```
addTwo(num)
```

```
    This function will add 2 to the input
```



# Getting Help

- Internet Search
- `dir([module])` – list function available
- `help([object])` – list documentation
- Add doc string when creating function

