Brief Tour of the Standard Library

Operating System Interface

The :mod:`os` module provides dozens of functions for interacting with the operating system:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 13); *backlink*

Unknown interpreted text role "mod".

```
>>> import os
>>> os.getcwd()      # Return the current working directory
'C:\\Python311'
>>> os.chdir('/server/accesslogs')   # Change current working directory
>>> os.system('mkdir today')   # Run the command mkdir in the system shell
0
```

Be sure to use the `import os` style instead of `from os import *`. This will keep :func:`os.open` from shadowing the built-in :func:`open` function which operates much differently.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 23); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 23); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 27)

Unknown directive type "index".

```
.. index:: builtin: help
```

The built-in :func:`dir` and :func:`help` functions are useful as interactive aids for working with large modules like :mod:`os`:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 29); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 29); *backlink*

Unknown interpreted text role "func".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 29); *backlink*

Unknown interpreted text role "mod".

```
>>> import os
>>> dir(os)
<returns a list of all module functions>
>>> help(os)
<returns an extensive manual page created from the module's docstrings>
```

For daily file and directory management tasks, the :mod:`shutil` module provides a higher level interface that is easier to use:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 38); *backlink*

Unknown interpreted text role "mod".

```
>>> import shutil
>>> shutil.copyfile('data.db', 'archive.db')
'archive.db'
>>> shutil.move('/build/executables', 'installdir')
'installdir'
```

File Wildcards

The :mod:`glob` module provides a function for making file lists from directory wildcard searches:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 53); *backlink*

Unknown interpreted text role "mod".

```
>>> import glob
>>> glob.glob('*.py')
['primes.py', 'random.py', 'quote.py']
```

Command Line Arguments

Common utility scripts often need to process command line arguments. These arguments are stored in the :mod:`sys` module's *argv* attribute as a list. For instance the following output results from running `python demo.py one two three` at the command line:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 66); *backlink*

Unknown interpreted text role "mod".

```
>>> import sys
>>> print(sys.argv)
['demo.py', 'one', 'two', 'three']
```

The :mod:`argparse` module provides a more sophisticated mechanism to process command line arguments. The following script extracts one or more filenames and an optional number of lines to be displayed:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 75); *backlink*

Unknown interpreted text role "mod".

```
import argparse

parser = argparse.ArgumentParser(
    prog='top',
    description='Show top lines from each file')
parser.add_argument('filenames', nargs='+')
parser.add_argument('-l', '--lines', type=int, default=10)
args = parser.parse_args()
print(args)
```

When run at the command line with `python top.py --lines=5 alpha.txt beta.txt`, the script sets `args.lines` to `5` and `args.filenames` to `['alpha.txt', 'beta.txt']`.

Error Output Redirection and Program Termination

The :mod:`sys` module also has attributes for *stdin*, *stdout*, and *stderr*. The latter is useful for emitting warnings and error messages to make them visible even when *stdout* has been redirected:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 99); *backlink*

Unknown interpreted text role "mod".

```
>>> sys.stderr.write('Warning, log file not found starting a new one\n')
Warning, log file not found starting a new one
```

The most direct way to terminate a script is to use `sys.exit()`.

String Pattern Matching

The :mod:`re` module provides regular expression tools for advanced string processing. For complex matching and manipulation, regular expressions offer succinct, optimized solutions:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 114); *backlink***
>
> Unknown interpreted text role "mod".

```
>>> import re
>>> re.findall(r'\bf[a-z]*', 'which foot or hand fell fastest')
['foot', 'fell', 'fastest']
>>> re.sub(r'(\b[a-z]+) \1', r'\1', 'cat in the the hat')
'cat in the hat'
```

When only simple capabilities are needed, string methods are preferred because they are easier to read and debug:

```
>>> 'tea for too'.replace('too', 'two')
'tea for two'
```

Mathematics

The :mod:`math` module gives access to the underlying C library functions for floating point math:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 136); *backlink***
>
> Unknown interpreted text role "mod".

```
>>> import math
>>> math.cos(math.pi / 4)
0.70710678118654757
>>> math.log(1024, 2)
10.0
```

The :mod:`random` module provides tools for making random selections:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 145); *backlink***
>
> Unknown interpreted text role "mod".

```
>>> import random
>>> random.choice(['apple', 'pear', 'banana'])
'apple'
>>> random.sample(range(100), 10)   # sampling without replacement
[30, 83, 16, 4, 8, 81, 41, 50, 18, 33]
>>> random.random()    # random float
0.17970987693706186
>>> random.randrange(6)    # random integer chosen from range(6)
4
```

The :mod:`statistics` module calculates basic statistical properties (the mean, median, variance, etc.) of numeric data:

> **System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 157); *backlink***
>
> Unknown interpreted text role "mod".

```
>>> import statistics
>>> data = [2.75, 1.75, 1.25, 0.25, 0.5, 1.25, 3.5]
>>> statistics.mean(data)
1.6071428571428572
>>> statistics.median(data)
1.25
>>> statistics.variance(data)
1.3720238095238095
```

The SciPy project <https://scipy.org> has many other modules for numerical computations.

Internet Access

There are a number of modules for accessing the internet and processing internet protocols. Two of the simplest are :mod:`urllib.request` for retrieving data from URLs and :mod:`smtplib` for sending mail:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 177); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 177); *backlink*

Unknown interpreted text role "mod".

```
>>> from urllib.request import urlopen
>>> with urlopen('http://worldtimeapi.org/api/timezone/etc/UTC.txt') as response:
...     for line in response:
...         line = line.decode()             # Convert bytes to a str
...         if line.startswith('datetime'):
...             print(line.rstrip())         # Remove trailing newline
...
datetime: 2022-01-01T01:36:47.689215+00:00

>>> import smtplib
>>> server = smtplib.SMTP('localhost')
>>> server.sendmail('soothsayer@example.org', 'jcaesar@example.org',
... """To: jcaesar@example.org
... From: soothsayer@example.org
...
... Beware the Ides of March.
... """)
>>> server.quit()
```

(Note that the second example needs a mailserver running on localhost.)

Dates and Times

The :mod:`datetime` module supplies classes for manipulating dates and times in both simple and complex ways. While date and time arithmetic is supported, the focus of the implementation is on efficient member extraction for output formatting and manipulation. The module also supports objects that are timezone aware.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 208); *backlink*

Unknown interpreted text role "mod".

```
>>> # dates are easily constructed and formatted
>>> from datetime import date
>>> now = date.today()
>>> now
datetime.date(2003, 12, 2)
>>> now.strftime("%m-%d-%y. %d %b %Y is a %A on the %d day of %B.")
'12-02-03. 02 Dec 2003 is a Tuesday on the 02 day of December.'

>>> # dates support calendar arithmetic
>>> birthday = date(1964, 7, 31)
>>> age = now - birthday
>>> age.days
14368
```

Data Compression

Common data archiving and compression formats are directly supported by modules including: :mod:`zlib`, :mod:`gzip`, :mod:`bz2`, :mod:`lzma`, :mod:`zipfile` and :mod:`tarfile`.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 234); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 234); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 234); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 234); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 234); *backlink*

Unknown interpreted text role "mod".

```
>>> import zlib
>>> s = b'witch which has which witches wrist watch'
>>> len(s)
41
>>> t = zlib.compress(s)
>>> len(t)
37
>>> zlib.decompress(t)
b'witch which has which witches wrist watch'
>>> zlib.crc32(s)
226805979
```

Performance Measurement

Some Python users develop a deep interest in knowing the relative performance of different approaches to the same problem. Python provides a measurement tool that answers those questions immediately.

For example, it may be tempting to use the tuple packing and unpacking feature instead of the traditional approach to swapping arguments. The :mod:`timeit` module quickly demonstrates a modest performance advantage:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 260); *backlink*

Unknown interpreted text role "mod".

```
>>> from timeit import Timer
>>> Timer('t=a; a=b; b=t', 'a=1; b=2').timeit()
0.57535828626024577
>>> Timer('a,b = b,a', 'a=1; b=2').timeit()
0.54962537085770791
```

In contrast to :mod:`timeit`'s fine level of granularity, the :mod:`profile` and :mod:`pstats` modules provide tools for identifying time critical sections in larger blocks of code.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 270); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 270); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 270); *backlink*

Unknown interpreted text role "mod".

Quality Control

One approach for developing high quality software is to write tests for each function as it is developed and to run those tests frequently during the development process.

The :mod:`doctest` module provides a tool for scanning a module and validating tests embedded in a program's docstrings. Test construction is as simple as cutting-and-pasting a typical call along with its results into the docstring. This improves the documentation by providing the user with an example and it allows the doctest module to make sure the code remains true to the documentation:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 284); *backlink*

Unknown interpreted text role "mod".

```
def average(values):
    """Computes the arithmetic mean of a list of numbers.

    >>> print(average([20, 30, 70]))
    40.0
    """
    return sum(values) / len(values)

import doctest
doctest.testmod()   # automatically validate the embedded tests
```

The :mod:`unittest` module is not as effortless as the :mod:`doctest` module, but it allows a more comprehensive set of tests to be maintained in a separate file:

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 302); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 302); *backlink*

Unknown interpreted text role "mod".

```
import unittest

class TestStatisticalFunctions(unittest.TestCase):

    def test_average(self):
        self.assertEqual(average([20, 30, 70]), 40.0)
        self.assertEqual(round(average([1, 5, 7]), 1), 4.3)
        with self.assertRaises(ZeroDivisionError):
            average([])
        with self.assertRaises(TypeError):
            average(20, 30, 70)

unittest.main()  # Calling from the command line invokes all tests
```

Batteries Included

Python has a "batteries included" philosophy. This is best seen through the sophisticated and robust capabilities of its larger packages. For example:

- The :mod:`xmlrpc.client` and :mod:`xmlrpc.server` modules make implementing remote procedure calls into an almost trivial task. Despite the modules names, no direct knowledge or handling of XML is needed.

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst, line 329); *backlink*

Unknown interpreted text role "mod".

System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-

resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 329); *backlink*

Unknown interpreted text role "mod".

- The :mod:`email` package is a library for managing email messages, including MIME and other RFC 2822-based message documents. Unlike :mod:`smtplib` and :mod:`poplib` which actually send and receive messages, the email package has a complete toolset for building or decoding complex message structures (including attachments) and for implementing internet encoding and header protocols.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 333); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 333); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 333); *backlink*

  Unknown interpreted text role "mod".

- The :mod:`json` package provides robust support for parsing this popular data interchange format. The :mod:`csv` module supports direct reading and writing of files in Comma-Separated Value format, commonly supported by databases and spreadsheets. XML processing is supported by the :mod:`xml.etree.ElementTree`, :mod:`xml.dom` and :mod:`xml.sax` packages. Together, these modules and packages greatly simplify data interchange between Python applications and other tools.

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 340); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 340); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 340); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 340); *backlink*

  Unknown interpreted text role "mod".

  System Message: ERROR/3 (D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main) (Doc) (tutorial)stdlib.rst, line 340); *backlink*

  Unknown interpreted text role "mod".

- The :mod:`sqlite3` module is a wrapper for the SQLite database library, providing a persistent database that can be updated

and accessed using slightly nonstandard SQL syntax.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst`, **line 349); *backlink***

Unknown interpreted text role "mod".

- Internationalization is supported by a number of modules including :mod:`gettext`, :mod:`locale`, and the :mod:`codecs` package.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst`, **line 353); *backlink***

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst`, **line 353); *backlink***

Unknown interpreted text role "mod".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\cpython-main\Doc\tutorial\(cpython-main)(Doc)(tutorial)stdlib.rst`, **line 353); *backlink***

Unknown interpreted text role "mod".