Python 3 Standard Library

Batteries Included

- Python 3 includes a large standard library of modules
  Together providing a huge number of useful attributes, functions and classes
- Next slides are a survey of a few modules and common usage
  We're only scratching the surface, See https://docs.python.org/3/library/index.html

Python Ecosystem

- Python has an even larger collection of 3rd party modules not part of the standard library
- The Python Package Index **PyPI** currently lists 89874 packages available for download
  - Ranging from simple utilities to complete software stacks for numerical/scientific computing to frameworks for development of web applications
    See https://pypi.org

`sys` module

- `sys.argv`
  list containing command line arguments

```
import sys

for num, arg in enumerate(sys.argv):
    print('arg {} is {!r}'.format(num, arg))
```

```
$ python3 printargs.py
arg 0 is 'printargs.py'

$ python3 printargs.py hello world 1.222
arg 0 is 'printargs.py'
arg 1 is 'hello'
arg 2 is 'world'
arg 3 is '1.222'
```

sys module

- `sys.stdin`
  `sys.stdout`
  `sys.stderr`

  The standard streams

```
import sys

sys.stdout.writelines(sorted(sys.stdin.readlines()))
```

```
$ cat cities.txt
New York
Los Angeles
Chicago
Philadelphia
Phoenix
Chicago
New York

$ python3 sortstdin.py < cities.txt
Chicago
Chicago
Los Angeles
New York
New York
Philadelphia
Phoenix
```

`os` module

Access operating system interfaces

- All functions may not be available on all OSes
- `os.getcwd()`
  Return current working directory as string
- `os.chdir()`
  Change current working directory
- `os.remove()`
  Remove a file

os.path module

Pathname manipulations

- os.path.join()
  Join pathname components using native separator
- os.path.exists()
  True if a pathname exists
- os.path.isfile()
  os.path.isdir()
  os.path.islink()

```
>>>> import os.path
>>>> os.path.join('workdir','subdir','data')
'workdir/subdir/data'

# On Windows
>>>> import os.path
>>>>
os.path.join(‘c:’,'workdir','subdir','data')
'workdir\subdir\data'
```

`glob` module

Pathname pattern expansion

`math` module

Mathematical functions (and constants)

```
>>> import math
>>> print(math.pi)
3.141592653589793

>>> print(math.e)
2.718281828459045

>>> math.sqrt(100)
10.0

>>> math.sqrt(40)
6.324555320336759

>>> math.log(128)
4.852030263919617

>>> math.log2(128)
7.0

>>> math.sin(math.pi/2)
1.0
```

`cmath` module

`math` with complex number support

- Extend `quadroot.py` program to support complex roots

```
>>> math.sqrt(-29)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: math domain error

>>> cmath.sqrt(-29)
5.385164807134504j

>>> type(cmath.sqrt(-29))
<class 'complex'>

>>> cmath.polar(3+7j)
(7.615773105863909, 1.1659045405098132)
```

`random` module

Generate pseudo-random numbers

- `random.randrange()`
  Random integer from a range
- `random.shuffle()`
  Random integer from a range
- `random.uniform()`
  `random.triangular()`
  `random.gauss()`

```
>>> random.randrange(-10,10)
8
>>> random.randrange(-10,10)
6
>>> random.randrange(-10,10)
-1
>>> random.randrange(-10,10)
-4
>>> random.randrange(-10,10)
…

>>> a = [1,2,3,4,5,6,7,8,9,10,11,12]
>>> random.shuffle(a)
>>> print(a)
[10, 9, 5, 8, 4, 7, 2, 12, 11, 1, 3, 6]

>>> random.choice('abcdefghijklm')
'h'
>>> random.choice('abcdefghijklm')
'a'
>>> random.choice('abcdefghijklm')
'c'
```

`statistics` module

High school statistics

```
>>> import statistics as stat
>>> rnlist = [random.randrange(-10,11) for i in range(20)]
>>> print(rnlist)
[8, -8, 7, 3, -5, 4, -1, 2, -7, -10, -6, -1, -3, 1, -2, 6,
10, 1, -9, -9]

>>> stat.mean(rnlist)
-0.95
>>> stat.median(rnlist)
-1.0
>>> stat.stdev(rnlist)
6.125657859407944
```

`datetime` module

Date and time calculations

```
>>>> import datetime as dt
>>>> t1 = dt.datetime(year=2015, month=2, day=28,
hour=1, minute=10, second=0 )
>>>> t2 = dt.datetime(year=2016, month=10, day=4,
hour=14, minute=00, second=0 )

>>>> print(t2-t1)
584 days, 12:50:00
```

`itertools` module

Iterator building blocks

```
>>>> import itertools
>>>> for i in itertools.product('abc','1234'):
....   print(i)
....
('a', '1')
('a', '2')
('a', '3')
('a', '4')
('b', '1')
('b', '2')
('b', '3')
('b', '4')
('c', '1')
('c', '2')
('c', '3')
('c', '4')

>>>> for i in itertools.permutations([1,2,3]):
....   print(i)
....
(1, 2, 3)
(1, 3, 2)
(2, 1, 3)
(2, 3, 1)
(3, 1, 2)
(3, 2, 1)
```

`decimal` module

Decimal floating point arithmetic

- Performs arithmetic the same way as people learn in school
- Decimal number represented exactly
  No binary floating point representation error
- Configurable precision, flags and traps

```
>>>> import decimal
>>>> D = decimal.Decimal

>>>> pone = D('0.1')
>>>> pthree = D('0.3')

>>>> pone + pone + pone == pthree
True
>>>> 0.1 + 0.1 + 0.1 == 0.3
False

>>>> decimal.getcontext()
Context(prec=28, rounding=ROUND_HALF_EVEN,
Emin=-999999, Emax=999999, capitals=1, clamp=0,
flags=[], traps=[InvalidOperation,
DivisionByZero, Overflow])
>>>> D(2).sqrt()
Decimal('1.414213562373095048801688724')
>>>> decimal.getcontext().prec=90
>>>> D(2).sqrt()
Decimal('1.414213562373095048801688724209698078569671875376948073176679737990732478462107038850387534')
```

`urllib.request` module

- Open URLs

subprocess module

- Subprocess management

```
>>> import subprocess as sp
>>> st1 = sp.Popen(['cat','cities.txt'], stdout=sp.PIPE)
>>> st2 = sp.Popen(['sort'], stdin=st1.stdout,
stdout=sp.PIPE)

>>> out, err = st2.communicate()
>>> print(out)
b'Chicago\nChicago\nLos Angeles\nNew York\nNew
York\nPhiladelphia\nPhoenix\n'
```