Core Python: Getting Started

MODULARITY

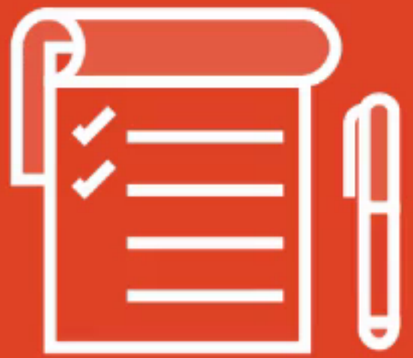
Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Overview



Reusable functions

Source code files called modules

Modules can be used from other modules

Importing modules

Programs or scripts

Python execution model

Make programs executable

Starting Code

```
from urllib.request import urlopen
story = urlopen('http://sixty-north.com/c/t.txt')
story_words = []
for line in story:
    line_words = line.decode('utf-8').split()
    for word in line_words:
        story_words.append(word)
story.close()
```

Open Text Editor

1. Python syntax highlighting
2. Four space indentation
3. UTF-8 encoding

Run Python from the Shell

```
$ cd corepy
$ python words.py
$
```

Run Changes

```
that
some
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
$
```

Importing Modules into the REPL

```
that
some
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
>>>
```

Defining Functions

```
>>> def square(x):
...     return x * x
...
>>> square(5)
25
>>> def launch_missiles():
...     print("Missiles launched!")
...
>>> launch_missiles()
Missiles launched!
>>>
```

Early Return

```
>>> def even_or_odd(n):
...     if n % 2 == 0:
...         print("even")
...         return
...     print("odd")
...
>>> w = even_or_odd(31)
odd
>>> w is None
True
>>>
```

```
def nth_root(radicand, n):
    return radicand ** (1/n)
```

```
>>> nth_root(16, 2)
4.0
>>> nth_root(27, 3)
3.0
>>>
```

```
def nth_root(radicand, n):
    return radicand ** (1/n)


def ordinal_suffix(value):
    s = str(value)
    if s.endswith('11'):
        return 'th'
    elif s.endswith('12'):
        return 'th'
    elif s.endswith('13'):
        return 'th'
    elif s.endswith('1'):
        return 'st'
    elif s.endswith('2'):
        return 'nd'
    elif s.endswith('3'):
        return 'rd'
    return 'th'


def ordinal(value):
    return str(value) + ordinal_suffix(value)


def display_nth_root(radicand, n):
    root = nth_root(radicand, n)
    message = "The " + ordinal(n) + " root of " \
              + str(radicand) + " is " + str(root)
    print(message)
```

- Calculate ordinal suffixes
- ST suffix for 1
- ND suffix for 2
- RD suffix for 3
- Define ordinal()
- Decomposition
- Display function
- implicit returns

```
def display_nth_root(radicand, n):
  root = nth_root(radicand, n)
  message = "The " + ordinal(n) + " root of " + str(radicand) + " is " + str(root)
  print(message)
```

```
>>> display_nth_root(64, 4)
The 4th root of 64 is 2.8284271247461903
>>>
```

Naming Special Functions

__feature__

Hard to prounounce!

dunder

Our way of pronouncing special names

A portmanteau of "double underscore"

Instead of "underscore underscore name underscore underscore" we'll say "dunder name"

Defining a Function

```
def fetch_words():
    story = urlopen('http://sixty-north.com/c/t.txt')
    story_words = []
    for line in story:
        line_words = line.decode('utf8').split()
        for word in line_words:
            story_words.append(word)
    story.close()

    for word in story_words:
        print(word)
```

```
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
>>>
$ python words.py
$
```

__name__

Specially named variable allowing us to detect whether a module is run as a script or imported into another module.

Print Name of Module

```
def fetch_words():
  story = urlopen('http://sixty-north.com/c/t.txt')
  story_words = []
  for line in story:
      line_words = line.decode('utf8').split()
      for word in line_words:
          story_words.append(word)
  story.close()

  for word in story_words:
      print(word)

print(__name__)
```

```
$ python
Python 3.7.4 (default, Oct 17 2019, 14:41:32)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> import words
words
>>> import words
>>>
$ python words.py
__main__
$
```

Import or Execute

```
from urllib.request import urlopen

def fetch_words():
  story = urlopen('http://sixty-north.com/c/t.txt')
  story_words = []
  for line in story:
      line_words = line.decode('utf8').split()
      for word in line_words:
          story_words.append(word)
  story.close()

  for word in story_words:
      print(word)

if __name__ == '__main__':
    fetch_words()
```

```
that
some
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
$
```

The Python Execution Model

`def` is a statement.

Top-level functions are defined when a module is imported or run.

Module, Script, or Program

Python module

Convenient import with API

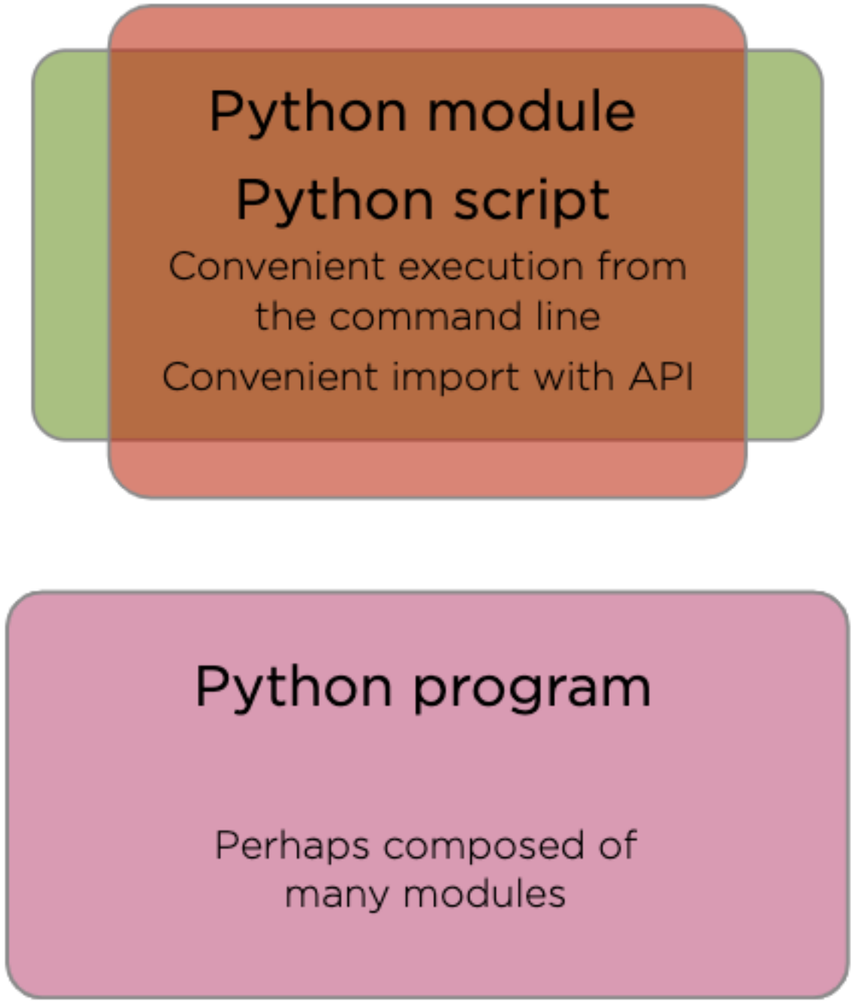
Python script

Convenient execution from the command line

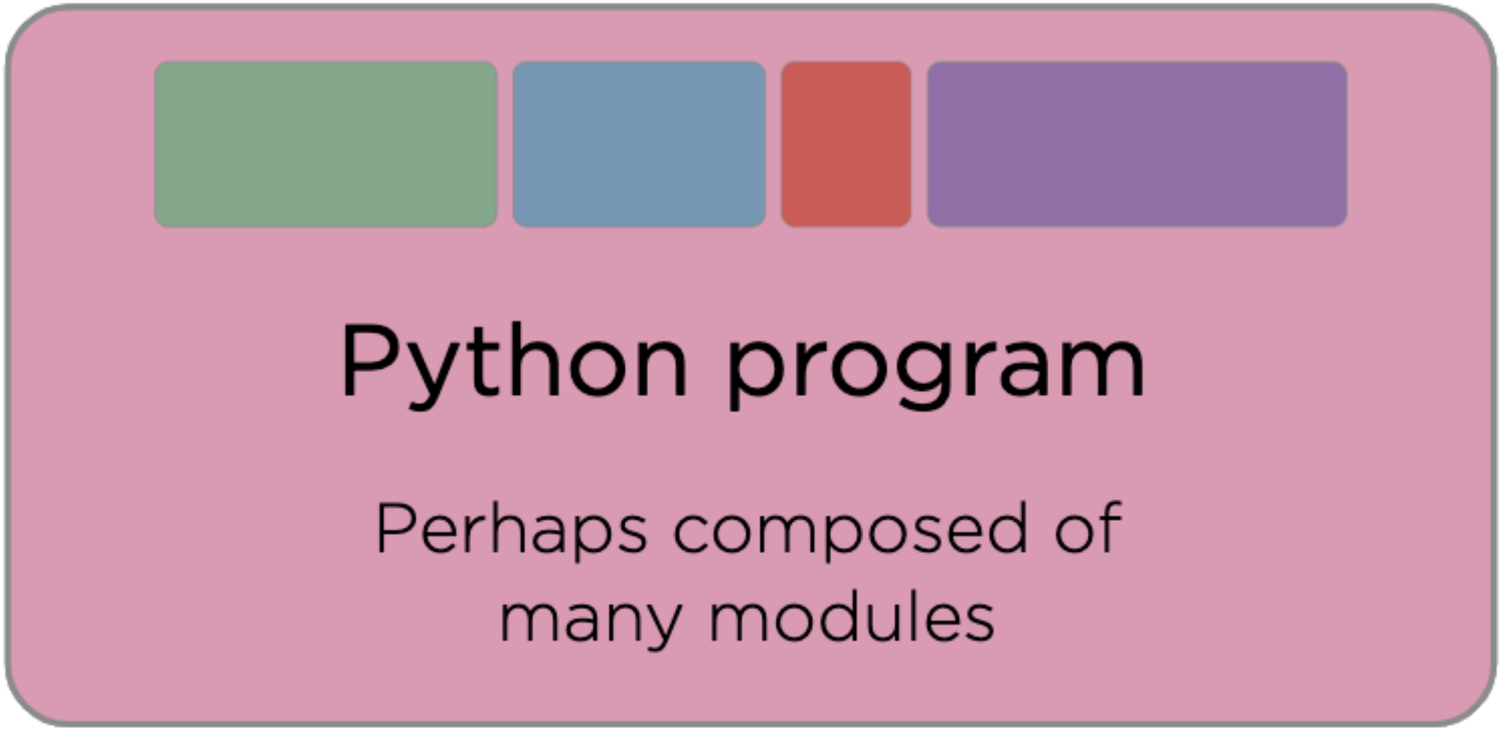
Python program

Perhaps composed of many modules

Module, Script, or Program



Module, Script, or Program



Command Line Arguments

Test in the REPL

```
t
r
i
n
g
s

a
r
e

i
t
e
r
a
b
l
e

t
o
o
>>>
```

Command Line and REPL

```
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
$ python
Python 3.7.4 (default, Oct 17 2019, 14:41:32)
[Clang 10.0.1 (clang-1001.0.46.4)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> from words import *
>>> main()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/Users/sixty-north/corepy/slide_spec/repl-2/words.py", line 22, in main
    url = sys.argv[1]
IndexError: list index out of range
>>>
```

Test in the REPL

```
that
some
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
>>>
```

Moment of Zen

Sparse is better than dense

Two between functions
That is the number of lines
PEP8 recommends

Docstrings

docstrings

Literal strings which document functions, modules, and classes.

They must be the first statement in the blocks for these constructs.

Sphinx

Tool to create HTML documentation from Python docstrings.

Docstrings

```
>>> from words import *
>>> help(fetch_words)
Help on function fetch_words in module words:

fetch_words(url)
    Fetch a list of words from a URL.

    Args:
        url: The URL of a UTF-8 text document.

    Returns:
        A list of strings containing the words from
        the document.
(END)
```

Docstrings

```
Help on module words:

NAME
    words - Retrieve and print words from a URL.

DESCRIPTION
    Usage:

       python3 words.py <URL>

FUNCTIONS
    fetch_words(url)
        Fetch a list of words from a URL.

        Args:
            url: The URL of a UTF-8 text document.

        Returns:
            A list of strings containing the words from
            the document.

    main(url)
        Print each word from a text document from at a URL.
:
```

Comments

Comments

Code is ideally clear enough without ancillary explanation

Sometimes you need to explain why your code is written as it is

Comments in Python start with # and extend to the end of the line

Make Script Executable

```
that
some
of
its
noisiest
authorities
insisted
on
its
being
received
for
good
or
for
evil
in
the
superlative
degree
of
comparison
only
$
```

Pylauncher

Pylauncher

1. Associated with `*.py` files
2. Executable is `py.exe` and is on the `PATH`
3. Parse the shebang and locate Python

Windows Command Prompt

```
> words.py http://sixty-north.com/c/t.txt
```

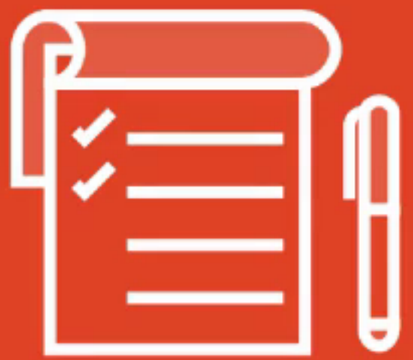
Windows PowerShell

```
PS> .\words.py http://sixty-north.com/c/t.txt
```

PEP 397

Describes PyLauncher

Summary



Python code is generally placed in *.py files

Execute modules by passing them as the first argument to Python

All top-level statements are executed when a module is imported

Define functions with the `def` keyword

Return objects from functions with the `return` keyword

`return` without an argument returns `None`, as does the implicit return

Summary

Use `__name__` to determine how a module is being used

`if __name__ == '__main__'` lets our module be executable and importable

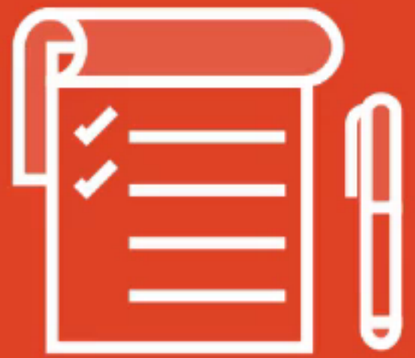
A module is executed once, on first import

`def` is a statement which binds code to a name

`sys.argv` contains command line arguments

Dynamic typing supports generic programming

Summary



Functions can have docstrings

`help()` can retrieve docstrings

Modules can have docstrings

Python comments start with #

Program loaders can use `#!` to determine which Python to run