Core Python: Getting Started

INSTALLING AND STARTING PYTHON

Austin Bingham
COFOUNDER - SIXTY NORTH

@austin_bingham

Robert Smallshire
COFOUNDER - SIXTY NORTH

@robsmallshire

Overview

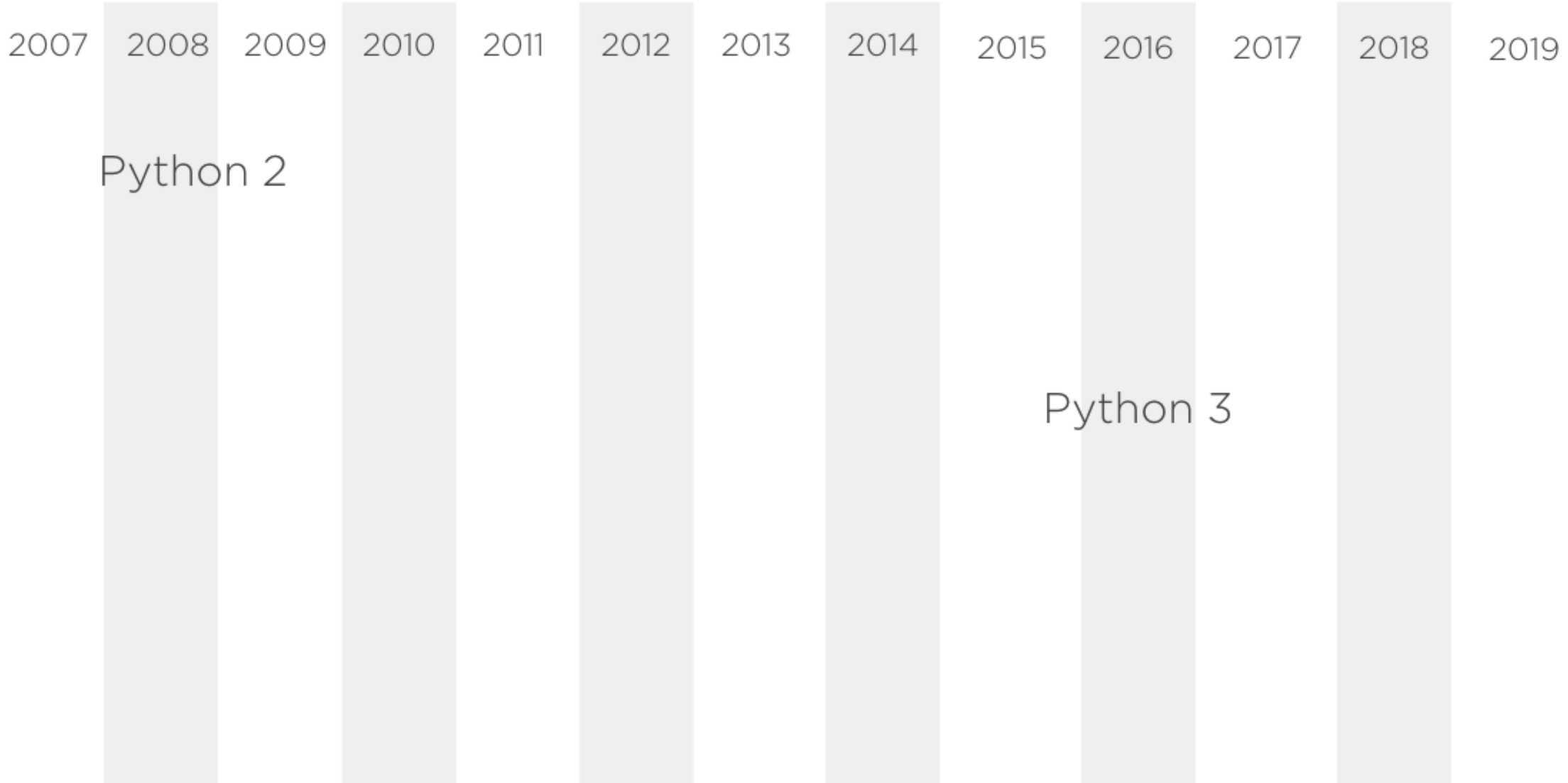
Install Python on your system

Write basic Python code

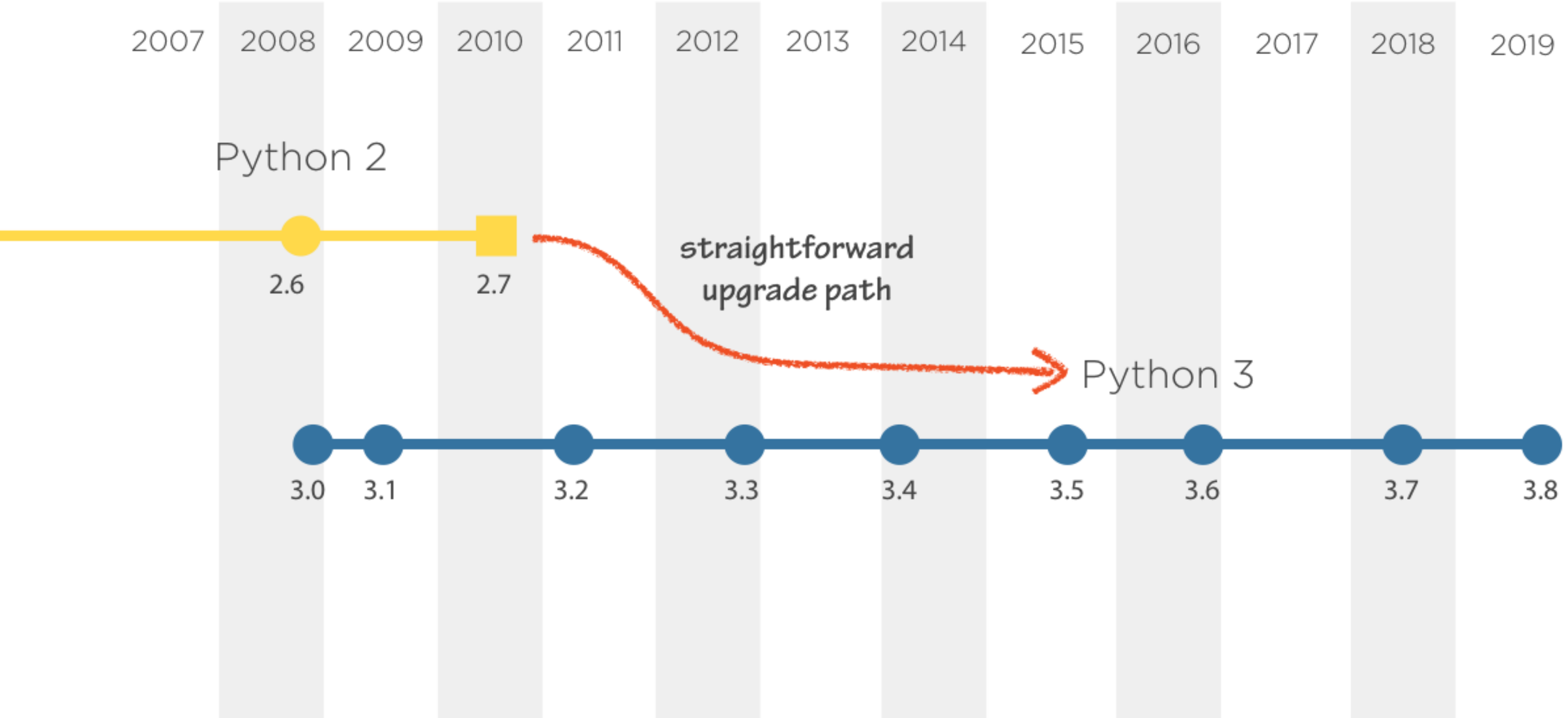
Get acquianted with Python programming culture

Never forget the origins of the name of the language

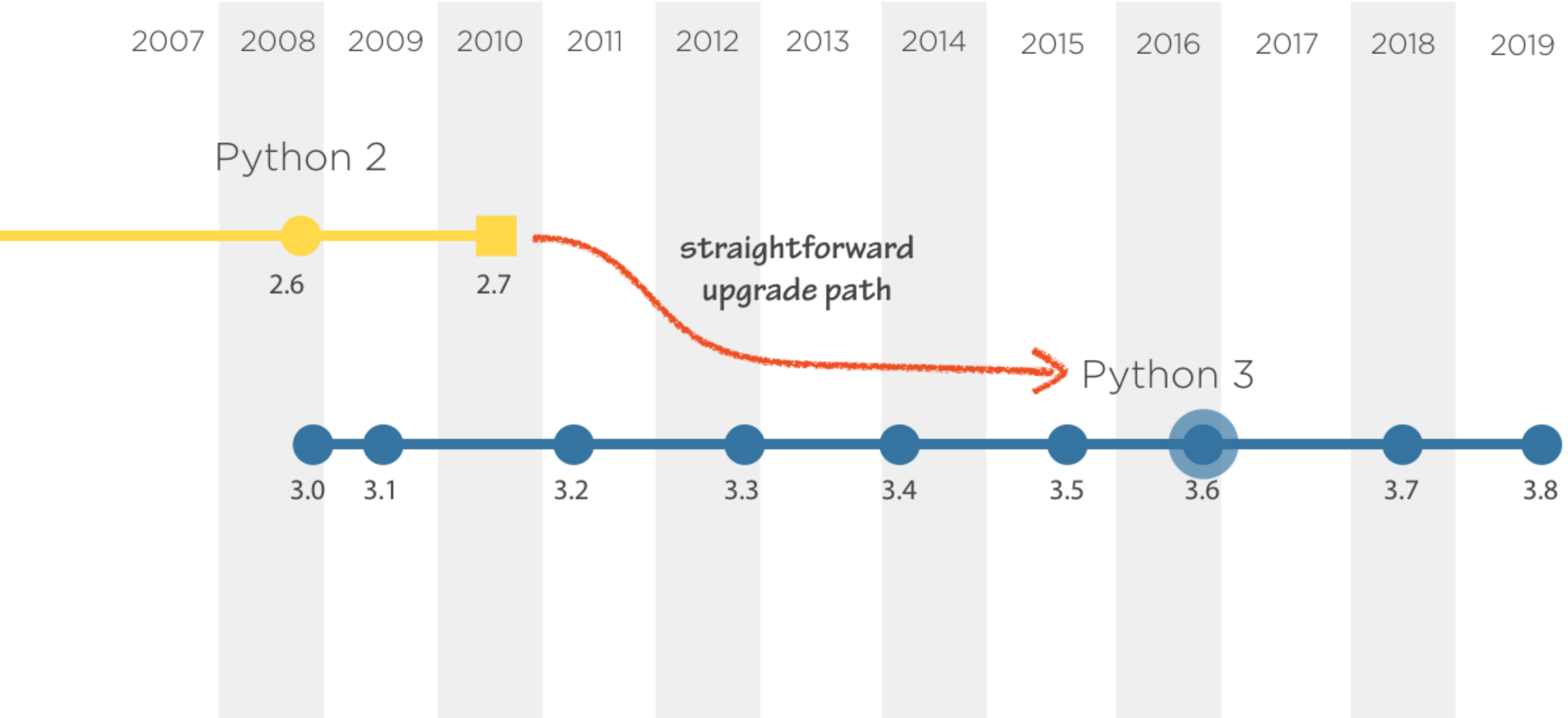
Python Release Timeline



Python Release Timeline



Python Release Timeline



Obtaining and Installing Python 3

Portable across Operating Systems

Windows

macOS

Linux

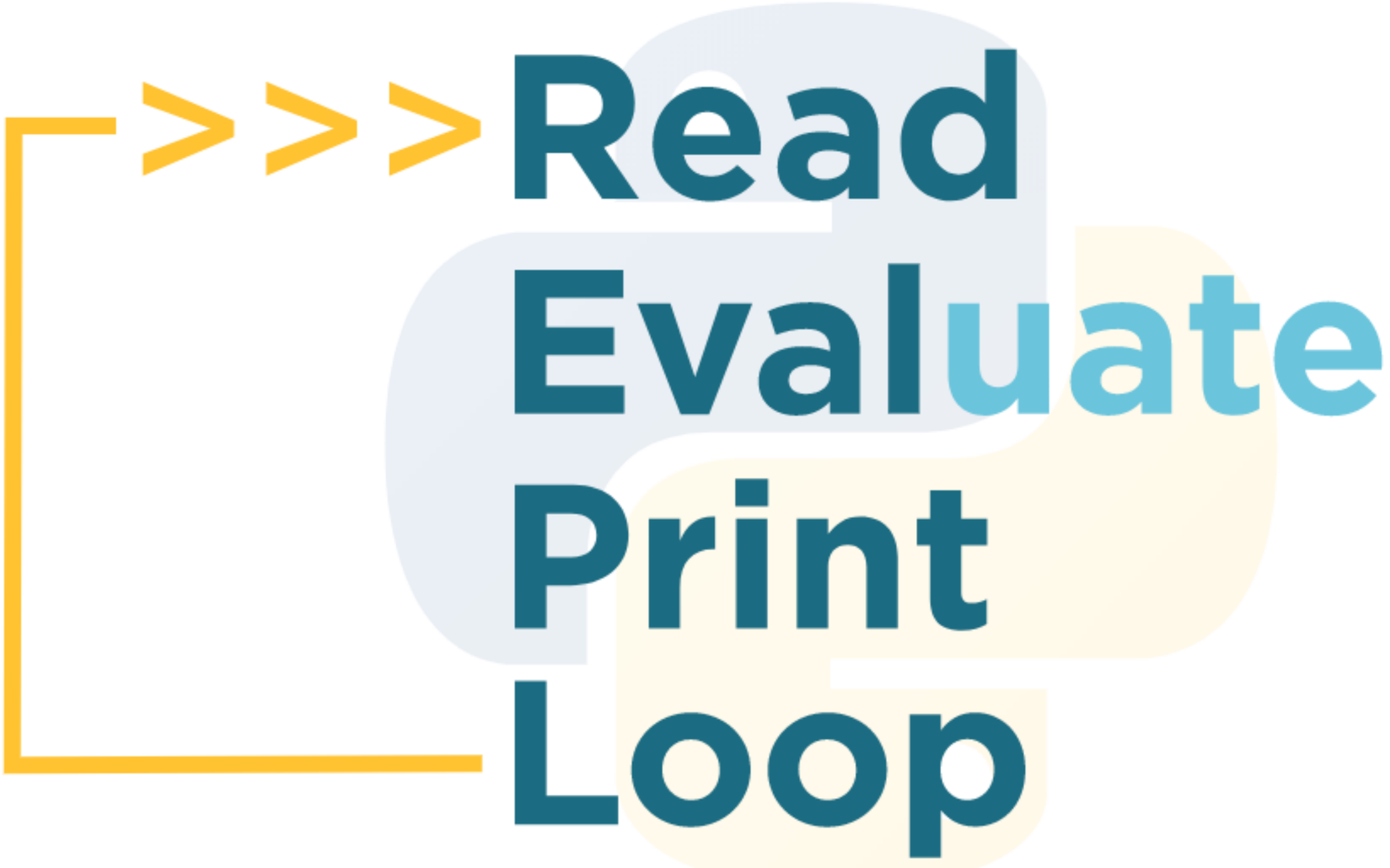
Installation on Windows

Installation on macOS

Installation on Linux

Interactive Python

Interactive Python




>>> REPL

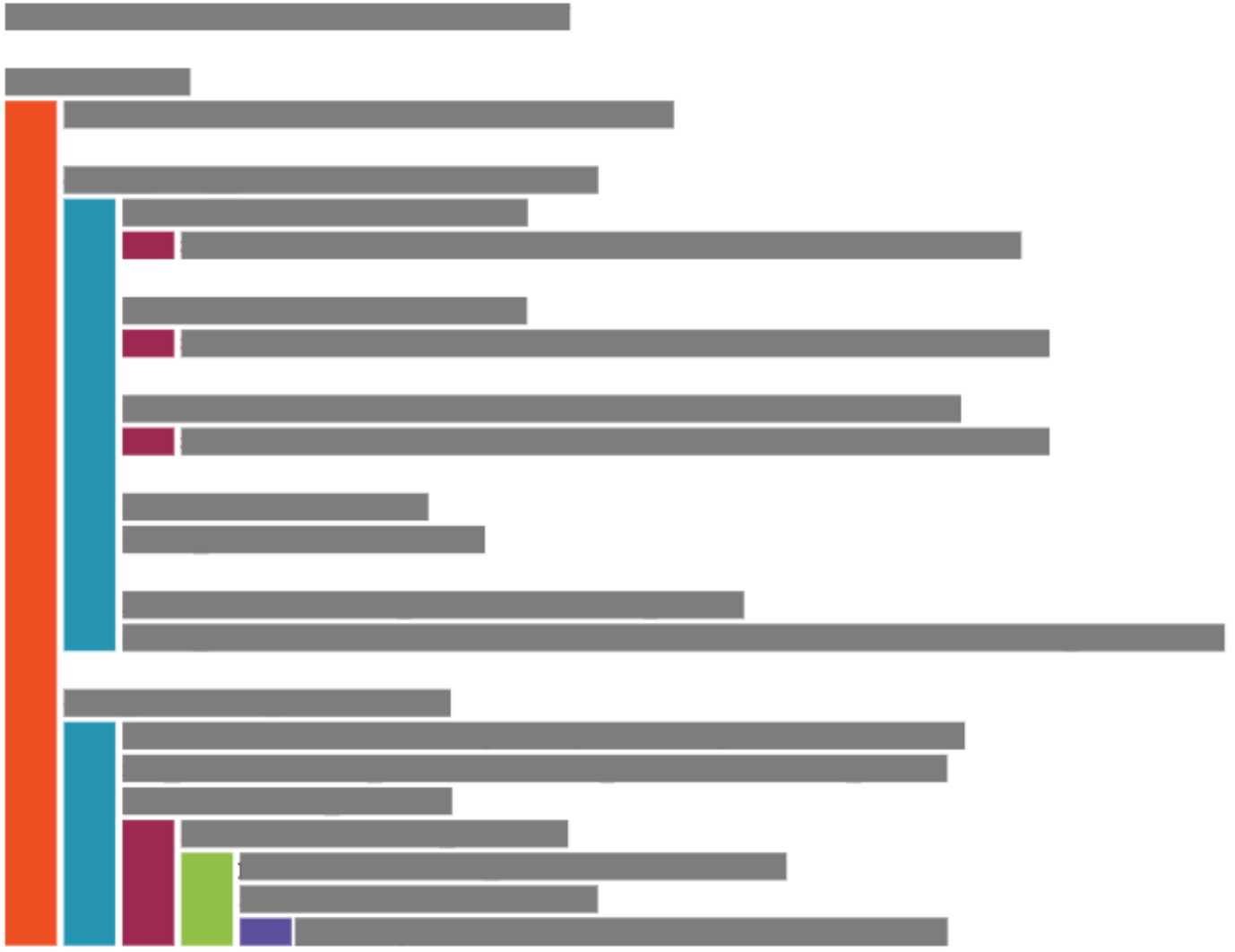
Printing in Python 2 and Python 3

```
print "Python 2"
```

Significant Whitespace

```
$ 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.
>>> for i in range(5):
...     x = i * 10
...     print(x)
...
0
10
20
30
40
>>>
```

Significant Whitespace



Significant Whitespace

Requires readable code

No clutter

Human and computer can't get out of sync

Significant Whitespace Rules

1. Prefer four spaces
2. Never mix spaces and tabs
3. Be consistent on consecutive lines
4. Only deviate to improve readability

Programming as Guido ~~intended~~ it indented

Python Culture

The Zen of Python

```
>>> import this
The Zen of Python, by Tim Peters

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
Special cases aren't special enough to break the rules.
Although practicality beats purity.
Errors should never pass silently.
Unless explicitly silenced.
In the face of ambiguity, refuse the temptation to guess.
There should be one-- and preferably only one --obvious way to do it.
Although that way may not be obvious at first unless you're Dutch.
Now is better than never.
Although never is often better than *right* now.
If the implementation is hard to explain, it's a bad idea.
If the implementation is easy to explain, it may be a good idea.
Namespaces are one honking great idea -- let's do more of those!
>>>
```

Moment of Zen

Readability Counts

Clarity matters
So readability makes
For valuable code

Using the Standard Library

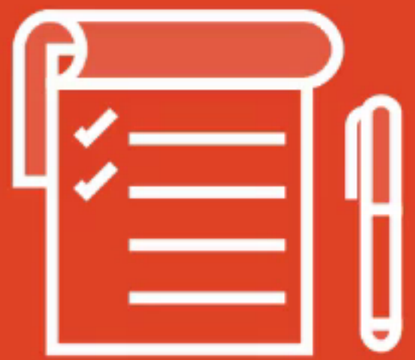
Batteries Included

The `import` keyword

```
import module_name
```

```
>>> from math import factorial
>>> factorial(n) / (factorial(k) * factorial(n - k))
10.0
>>> from math import factorial as fac
>>> fac(n) / (fac(k) * fac(n - k))
10.0
>>> fac(n) // (fac(k) * fac(n - k))
10
>>> 2**31 - 1
2147483647
>>> fac(13)
6227020800
>>> fac(13) > 2**31 - 1
True
>>> n = 100
>>> k = 2
>>> fac(n) // (fac(k) * fac(n - k))
4950
>>> fac(n)
933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000
>>> len(str(fac(n)))
158
>>>
```

Summary



Download and install Python

Start the Python REPL

Evaluate simple expressions

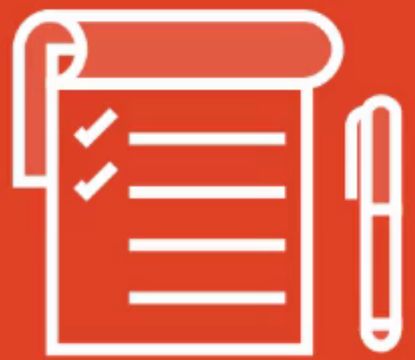
The role of underscore in the REPL

How to use `print()`

- Output is a side-effect

Exiting the REPL

Summary



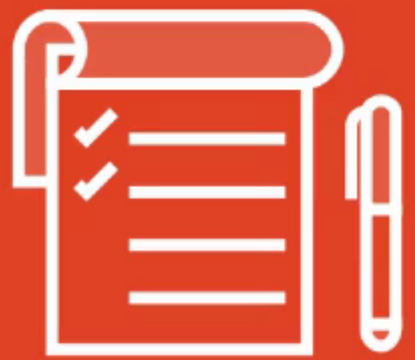
Significant whitespace

- Colons and indentation
- Advantages of significant whitespace
- Rules for indentations

Python culture

- The Zen of Python
- "Readability Counts"

Summary



Importing standard library modules

- `import from`
- `from module import name`
- `from module import name as name2`

Using Python's help system

`math.factorial`