Introduction to Python

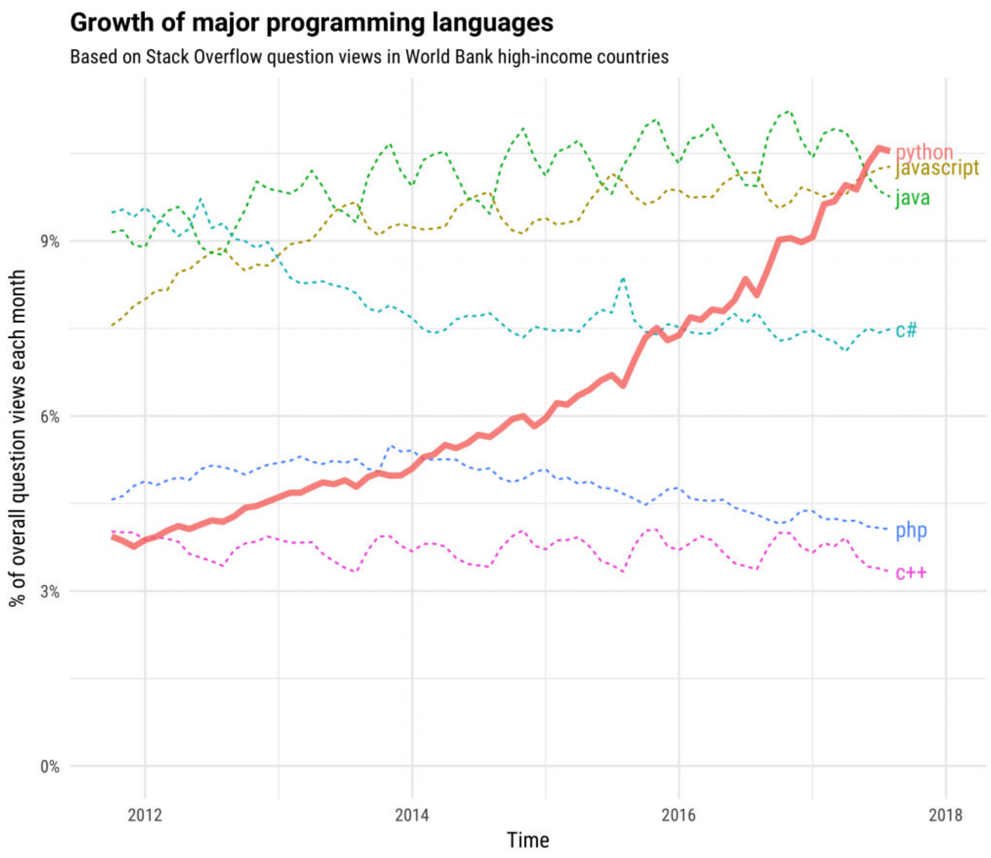
Objectives:

- What is Python?
- Why learn Python?
- Python Syntax
- Lovely Jupyter Notebook

1. What is Python

Python is an interpreted, object-oriented, high-level programming language.

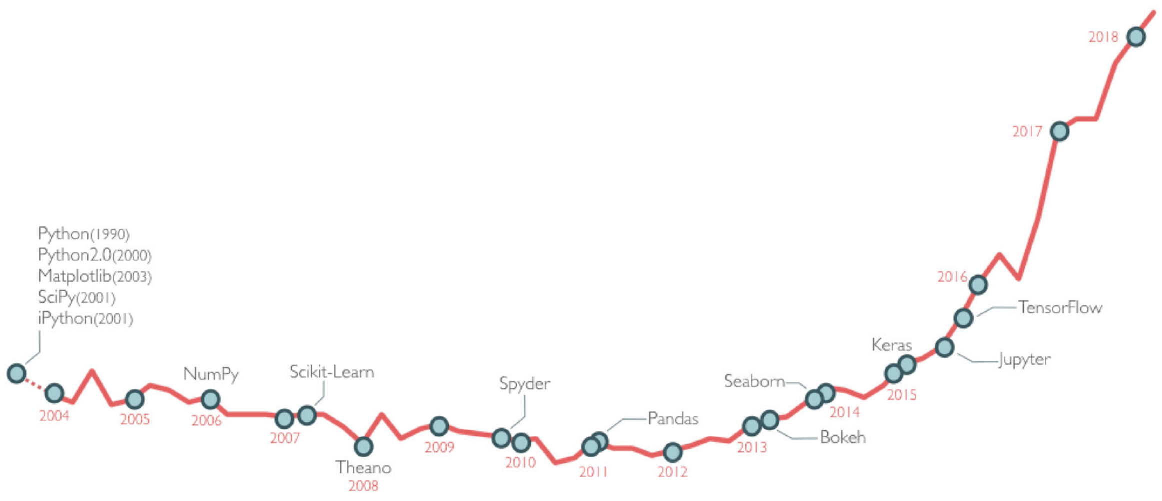
- Interpreted instead of compilation
- Cross-Platform on all major computer OS including hobbyists
- Object-oriented for modularity and code reuse
- High-level and good for rapid application development



Reference: https://hackernoon.com/how-is-python-different-from-other-programming-languages-63311390f8dd (https://hackernoon.com/how-is-python-different-from-other-programming-languages-63311390f8dd)

1.1 Advantages

- Simple and Easy to Learn
- Free and Open Source
- Portable and Extensible
- Many 3rd-Party Frameworks and Tools



Reference: https://medium.com/@atillaguzel/popularity-of-data-science-python-and-pythons-major-libraries-f7146e202e5d (https://medium.com/@atillaguzel/popularity-of-data-science-python-and-pythons-major-libraries-f7146e202e5d)

1.2 Disadvantages

- Python is not as fast, especially compared to compiled languages
- Python does not scale well with multiprocessor or multicore systems

1.3 Applications

- Web and Internet Development
- Scientific and Numeric Computing
- Scripting small programs for simple task automation
- Desktop GUI

Reference: https://www.python.org/about/apps/ (https://www.python.org/about/apps/)

1.4 Move on to Python 3

Major versions

- Python 1.0.0 was released on 26 Jan 1994
- Python 2.0 was released on 16 Oct 2000
- Python 3.0 was released on 3 Dec 2008

There are 2 major versions in used, version 2.x and 3.x.

- Version 3.x is not backward compatible with version 2.x
  - Legacy libraries/code must be re-written
- Version 2 will be End Of Life (EOL) in January 2020
  - no further updates nor bugfixes

You can check out the differences between version 2 and 3 in following site.

Reference: https://www.guru99.com/python-2-vs-python-3.html (https://www.guru99.com/python-2-vs-python-3.html)

2. Python Syntax

The default window of IDLE is an interactive Read-Eval-Print-Loop (REPL) environment, where user can type command directly. The interpreter will

- Reads the command entered by user
- Evaluate and execute the command
- Print the output (if any) to the console
- Loop back and repeat the process

2.1 Hello World

As a great programmer tradition, we always start learning new programming language by saying **`Hello World`** .

Exercise: Print out `Hello World` .

In [1]:
```
print('Hello World!')
```

```
Hello World!
```

2.2 Variables

A **variable** represents an entity which holds a value.

- Variables in Python don’t require declaration
- Variables must be initialized before use.

Exercise: Create 3 variable, `x = 1` , `y = 2.0` and `z = "hello world"` .

In [2]:
```
x = 1
y = 2.2
z = "hello world"
print(x, y, z)
```

```
1 2.2 hello world
```

Question:

Given a variable, how do you know its data type?

In [3]:
```
type(x)
```

Out[3]:
```
int
```

Exercise: Check data type of variable `z` .

In [4]:
```
type(z)
```

Out[4]:
```
str
```

2.3 Comment

Comments are non-executable part of any programming code.

- They are commonly used in codes for documentation.
- Writing comments is a good programming practice.

Python use `#` character.

- Every line of comment in Python must begin with a `#` character.
- In-line comments are comment behind a statement.

Exercise:

Try out following commands.

In [5]:
```
# this is a comment
x = 3.14       # pi value
print(x)
```

```
3.14
```

2.4 Indentation

Many high-level programming languages, e.g. C and Java, use braces `{ }` to mark a block of code.

Python does it via indentation.

- Statements in a Python code block have same indentation.
- For example, body of a function or a loop

Exercise:

What's the error message when following code runs?

In [6]:
```
msg = 'hello'
  name = 'world'
```

```
  File "<ipython-input-6-5e8c33ea00f9>", line 2
    name = 'world'
    ^
IndentationError: unexpected indent
```

2.5 Python Identifiers

Python Identifiers are user-defined names to represent a variable, function, class, module or any other object.

Guidelines for Creating Identifiers

- Identifier name can contain following charaters
  - a sequence of letters either in lowercase (a to z) or uppercase (A to Z)
  - digits (0 to 9)
  - underscore (_)
- Identifier name cannot begin with digits
- Special characters are not allowed

Exercise: Type following commands line by line. Press `CTRL + ENTER` after each line.

In [8]:
```
message = 'ok'
_message = 'ok'
```

In [9]:
```
1message = 'not ok'
message@ = 'not ok'
```

```
  File "<ipython-input-9-9d330a14dc9b>", line 1
    1message = 'not ok'
     ^
SyntaxError: invalid syntax
```

2.6 Python Keywords

Keywords are special words which are reserved and have a specific meaning. Python has a set of keywords that cannot be used as variables in programs.

Exercise: List of Python keywords

In [10]:
```
import keyword
print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break',
'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'finally', 'fo
r', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'no
t', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

Exercise: Keywords should NOT be used as identifiers because they are reserved.

In [11]:
```
for = 1
True = 1
```

```
  File "<ipython-input-11-24d7e5e21913>", line 1
    for = 1
        ^
SyntaxError: invalid syntax
```

2.8 Ask for Help

Python provides a useful `help()` function which can be used to find out more information on any identifier.

Exercise: Find out more about `print()` function.

In [12]:
```
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.
```

2.9 Get Help in Jupyter Notebook

Jupyter Notebook provides much convenient ways to get help.

- Use `?` to get documentation on a command or information on an object
- Combine `?` and `*` to find attributes of an object

Using ?

Exercise: Find out more about `print()` function.

In [ ]:
```
print?
```

Exercise: Find out more about an object, e.g. an integer `x` .

In [ ]:
```
x = 10
x?
```

Using *?

Exercise: Find out what methods starting with `s` are available for a String variable, `y = 'hello world'` .

In [13]:
```
y = '    hello world    '
y.s*?
```

Check out the method `strip()` on its purpose and how to use it.

In [14]:
```
y.strip()
```

Out[14]:
```
'hello world'
```

Recap

What are the differences between Python and other programming languages?

```

```

What's Jupyter Notebook?

```

```