Applied Data Science with Python

Istvan Lengyel

ilengyel@eit.ac.nz

Senior Lecturer, EIT

Kamini Garg
kaminigarg32@gmail.com

Course Goals

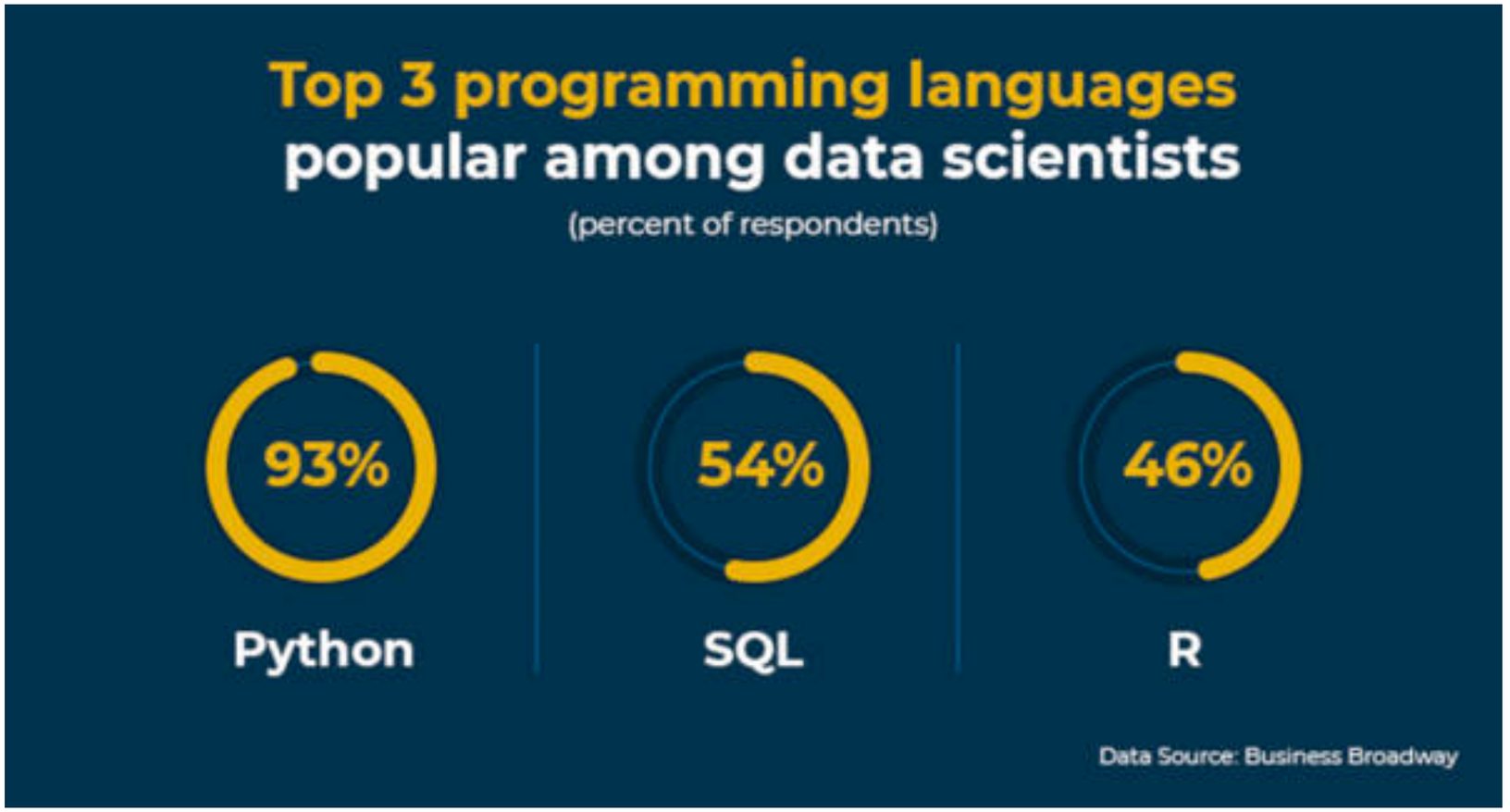
- Python basics
- To learn how Python can be used for:
  1. Data analysis
  2. Data exploration
  3. Machine Learning
- Learn by implementing hands-on examples & end-to-end pipelines

Why Python?

- Very High-level Language
  - ✓ Provides many data structures like lists, dictionary
  - ✓ Shorter code as compare to other languages
- Powerful libraries to perform complex data science tasks
  - ✓ Numpy, Pandas, Scipy, Scikit-learn …
- Well-documented and open source
- Widely adopted in Scientific community
- Very large and collaborative developer community

Many more…

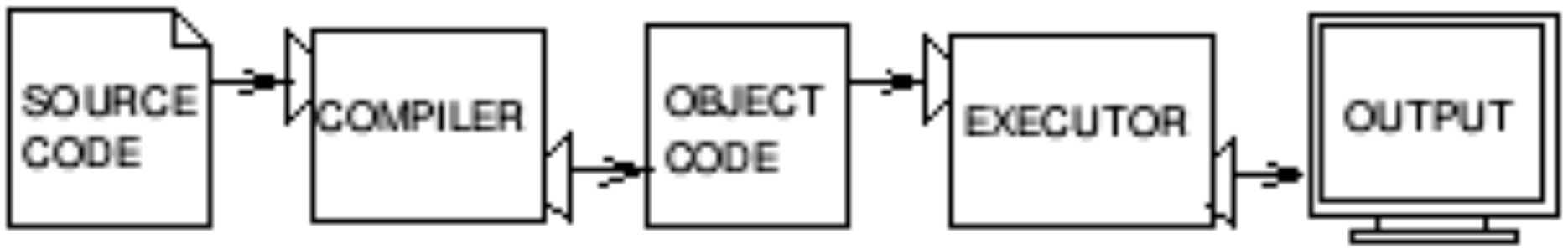
First choice for Data Scientists



The survey was carried out by Kaggle on nearly 24,000 data professionals, wherein 3 out of 4 respondents recommended aspiring data scientists to begin their learning journey with Python (source: kdnuggets).

How it works ?

High level language ----> Low level language
Interpreters, Compilers



Python Installation

- In this course, we will use Python 3.X & Google Colab, Google drive.
- To minimise any deprecation errors due to versioning , we will avoid using an IDE environment.
- Google Colab and Google Drive will require a google account, we will walk through this processes now by going to:

  https://colab.research.google.com/notebooks/intro.ipynb
- If you do not have a Google drive account please create one:

  https://drive.google.com
- You will use this for saving your work.

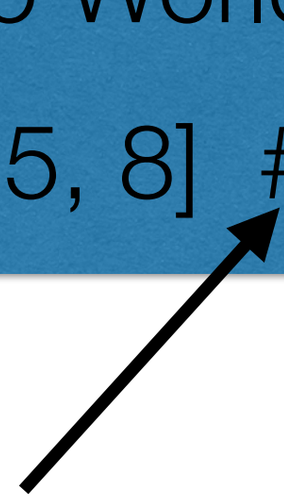
Python Variables

A variable is a name that refers to a value.

```
x = 1.51535

greeting = 'Hello World'

num = [1, 1, 2, 3, 5, 8]  # list
```



Refers comment

Legal Variable Names

- Have to begin with letters.
- 76trombones = 'big parade' Invalid
- Only underscore can be used in special characters.

Try it out :

class = 'how are you?'

Types

```
>>> type('Hello, world!')
<class 'str'>
```

String of characters

```
>>> type(42)
<class 'int'>
```

Integer

```
>>> type(3.141456)
<class 'float'>
```

Floating point number

Arithmetic Operators

```
5+12

12*4.514

(84-3+6)/5

'Hello' + ‘OTH'

‘spam’*3
```

String arithmetic operation: only **+** (also called concatenation) and ***** (for repetition) can be performed

Comparison Operators

```
x == 1 # check for equality
y != 'hello' # check for inequality
2 > 2 # False
2 >= 2 # True
2.1 < z < 5.4 # chained inequalities
```

Boolean Operations

x = 5, y = ’OTH’
x == 5 or y == ’OTH’
x > 0 and y != ’OTH’
y = 0
x or y
x and y
not y

Python uses **or, not, and** conveniently instead of **||, !, &&**

Conditional Statements: If

Gives the ability to check the condition and change the behavior of the program accordingly.

```
if x > 0:
  print ('x is positive’)
if not(x > 0):
  print ('x is negative')
if x == 0:
  print(‘x is zero’)
```

if/else

```
x=Insert_a_value
if x%2 == 0:
    print ('x is even')
else:
    print ('x is odd')
```

Note: The % symbol refers to modulo, which refers to the remainder from a division operation

While Loop

The while loop repeatedly executes a task while a condition is true.

```
number = 1

while number <= 10 :

    print(number\n)
    number += 1


print('Done!')
```

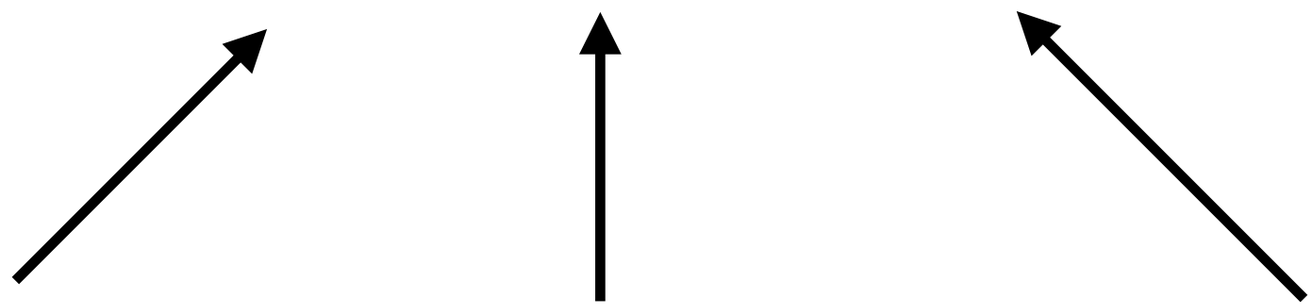
Break / Continue

Using break and continue

- break: used to jump out of the loop.
- Example: take user input till they type “Done”!!
- Continue is opposite of break :)

for Loop

```
for (i = 0; i < n; i = i + 1)
```



Initialize index variable	Condition to continue running the code	Increment at the end of each loop

- Python uses ***for*** differently than Matlab, C++, ...
- ***for*** is used to iterate over elements in an “object”
- This is one reason why Python is easy and powerful

for Loop

```
for x in range(0, 3):
    print('The number is:', x)
```

Output

```
0
1
2
```

More about range

```
>>> range(10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> range(1, 11)
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> range(0, 30, 5)
[0, 5, 10, 15, 20, 25]
>>> range(0, 10, 3)
[0, 3, 6, 9]
>>> range(0, -10, -1)
[0, -1, -2, -3, -4, -5, -6, -7, -8, -9]
>>> range(0)
[]
>>> range(1, 0)
[]
```

Strings

```
>>> 'spam and eggs'
'spam and eggs'

>>> hello = 'Greetings!'
>>> hello
'Greetings!'

>>> print(hello)
Greetings!

>>> print(hello + ' How do you do?')
Greetings! How do you do?
```

Strings : len function

```
>>> fruit = 'banana'
>>> length = len(fruit)
>>> print(length)
>>> 6
>>> print(fruit[length])
>>> ???
>>> print(fruit[-1]) ???
```

String Slices

A piece/segment of string is called slice.

```
>>> s = 'Monty Python'
>>> print s[0:5]
Monty
>>> print s[6:12]
Python
```

String are immutable

Cannot change the existing string.

```
>>> greeting = 'Hello, world!'
>>> greeting[0] = 'J'
TypeError: object does not support item assignment
```

```
>>> greeting = 'Hello, world!'
 >>> new_greeting = 'J' +  greeting[1: ]
>>> print(new_greeting)
Jello, world!
```

Using input in script

input function is used to take values from user.

```
>>> name = input('Enter your name please: ')
>>> print('Hello', name)
```

Exercises

1. Approximate the square root of 2 in precision of 0.01 without using math module.
2. Traverse the following string using loop ?

test = 'This is a great opportunity to examine the value of strings.’