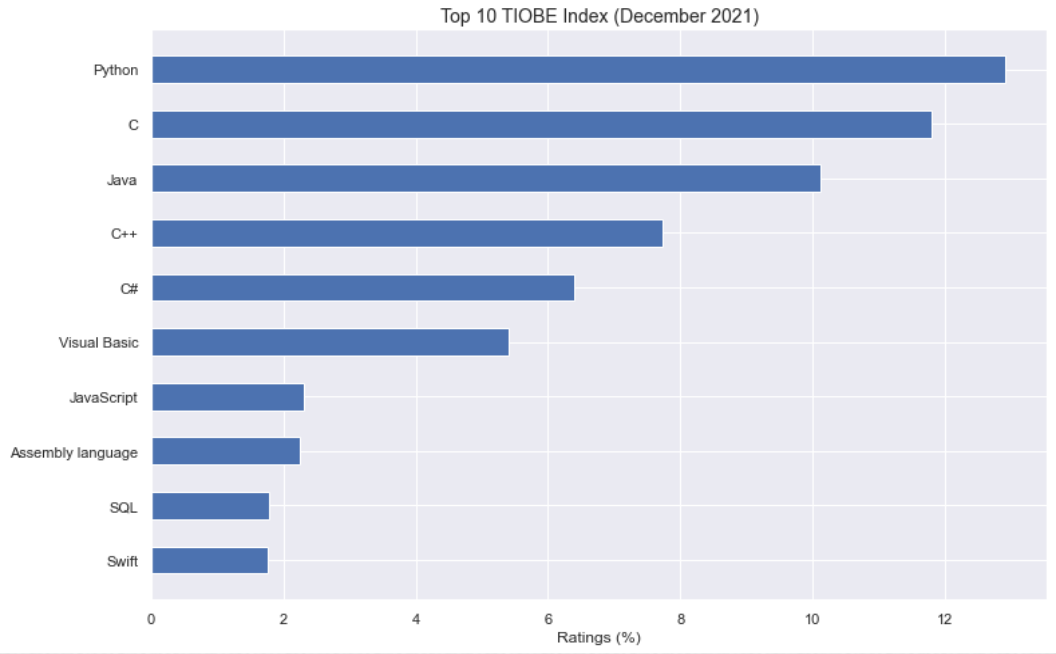
Predictive Analytics Foundations

Lecture 2

Lecture 2

- **History of Python**
- Installing & Running Python
- IDE
- Hands-on

Data Science Programming



Brief History of Python

Created in 1991 by Guido van Rossum (now at Google)	Useful as a **scripting language**	Used by:
• Named for Monty Python	• **script**: A small program meant for one-time use • Targeted towards small to medium sized projects	• Google, Yahoo!, Youtube • Many Linux distributions • Games and apps (e.g. Eve Online)

Python's Benevolent Dictator For Life

Lecture 2

- History of Python
- **Installing & Running Python**
- IDE
- Hands-on

Installing Python

Windows:

- Download Python from http://www.python.org
- Install Python.
- Run **Idle** from the Start Menu.

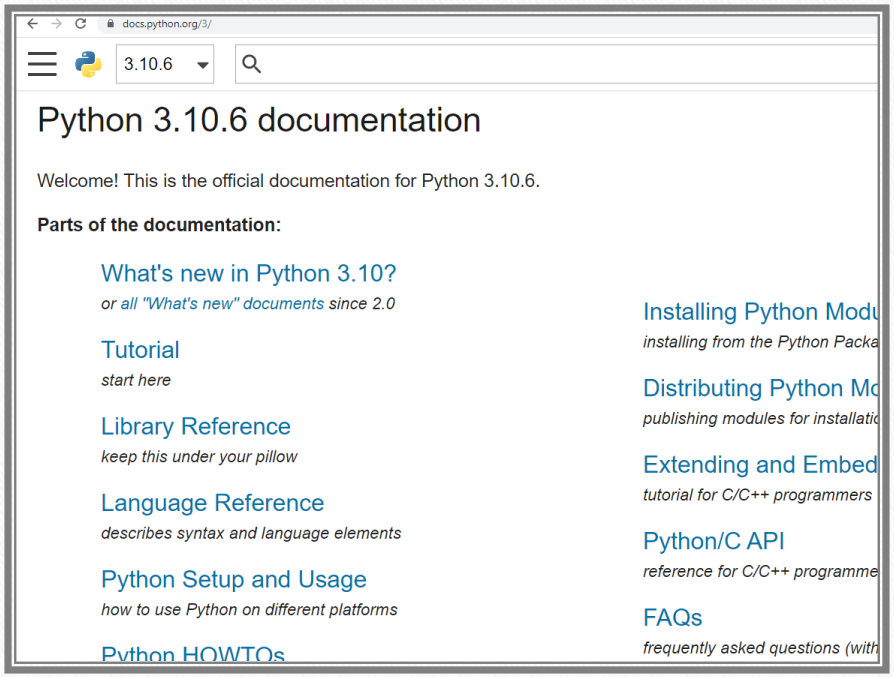
Mac OS X:

- Python is already installed.
- Open a terminal and run python or run Idle from Finder.

Linux:

- Chances are you already have Python installed. To check, run python from the terminal.
- If not, install from your distribution's package system.

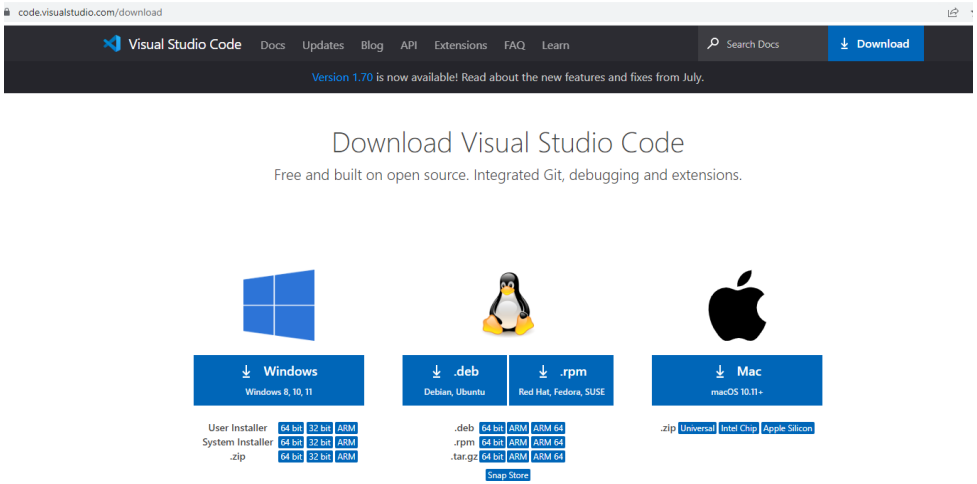
http://docs.python.org/



The Python tutorial is good!

Lecture 2

- History of Python
- Installing & Running Python
- **IDE**
- Hands-on



Installing VS Code

What about Jupyter?

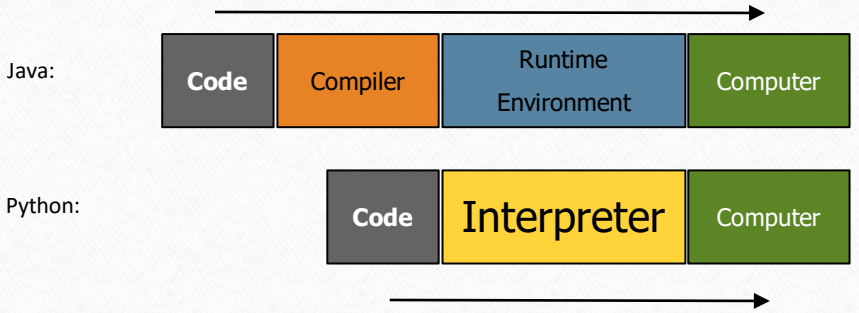
Google Colab

Lecture 2

- History of Python
- Installing & Running Python
- IDE
- **Hands-on**

Interpreted Languages

- **Interpreted**
- Not compiled like Java
- Code is written and then directly executed by an **interpreter**
- Type commands into interpreter and see immediate results



The Python Interpreter

Typical Python implementations offer both an interpreter and compiler

Interactive interface to Python with a read-eval-print loop

- >>> **map(square, [1, 2, 3, 4])**
- [1, 4, 9, 16]
- >>>

Lecture 2

- History of Python
- Installing & Running Python
- IDE
- **Hands-on**

A Code Sample (in IDLE)

```
x = 34 - 23                 # A comment.
y = “Hello”                 # Another one.
z = 3.45
if z == 3.45 or y == “Hello”:
    x = x + 1
    y = y + “ World”        # String concat.
print(x)
print(y)
```

Enough to Understand the Code

- Indentation matters to code meaning
  - *Block structure indicated by indentation*
- First assignment to a variable creates it
  - *Variable types don't need to be declared.*
  - *Python figures out the variable types on its own.*
- Assignment is = and comparison is ==
- For numbers + - * / % are as expected
  - *Special use of + for string concatenation and % for string formatting (as in C's printf)*
- Logical operators are words (and, or, not) *not* symbols
- The basic printing command is print

Basic Data Types

- Integers (default for numbers)

```
z = 5 / 2  # Answer 2, integer division
```

- Floats

```
x = 3.456
```

- Strings
  - Can use “” or ‘’ to specify with `“abc” == ‘abc’`
  - Unmatched can occur within the string: `“matt’s”`
  - Use triple double-quotes for multi-line strings or strings than contain both ‘ and “ inside of them:
    `“““a‘b“c”””`

White Space

- Whitespace is meaningful in Python: especially indentation and placement of newlines
- Use a newline to end a line of code
  - Use \ when must go to next line prematurely
- No braces {} to mark blocks of code, use *consistent* indentation instead
  - First line with *less* indentation is outside of the block
  - First line with *more* indentation starts a nested block
- Colons start of a new block in many constructs, e.g. function definitions, then clauses

Comments

- Start comments with #, rest of line is ignored
- Can include a “documentation string” as the first line of a new function or class you define
- Development environments, debugger, and other tools use it: it’s good style to include one

```
def fact(n):
    """fact(n) assumes n is a positive integer and returns facorial of n."""
    assert(n>0)
    return 1 if n==1 else n*fact(n-1)
```