Welcome to
Intro to Script Programming/Python

Your Instructor:

BIBEK UPADHAYAY
Adjunct Instructor

Email: bupadhayay@newhaven.edu
Website: ibibek.com

Office Hours: Apt. via Zoom.

Subject: **[S2022_Python]**

Brief history of python

- Invented in the Netherlands, early 90s by Guido van Rossum

- Named after Monty Python

- Open sourced from the beginning, managed by *Python Software Foundation*

- Considered a scripting language, but is much more

- Scalable, **object oriented** and functional from the beginning

- Used by Google from the beginning

"Python is an experiment in how much freedom programmers need. Too much freedom and nobody can read another's code; too little and expressive-ness is endangered. ."
-Guido van Rossum

Where is Python now?

Rank	Language	Type			Score
1	Python	🌐	💻	⚙	100.0
2	Java	🌐 📱	💻		95.4
3	C	📱	💻	⚙	94.7
4	C++	📱	💻	⚙	92.4
5	JavaScript	🌐			88.1
6	C#	🌐 📱	💻	⚙	82.4
7	R		💻		81.7
8	Go	🌐	💻		77.7
9	HTML	🌐			75.4
10	Swift	📱	💻		70.4

IEEE SPECTRUM

Worldwide, Jan 2022 compared to a year ago:

Rank	Change	Language	Share	Trend
1		Python	28.74 %	-1.8 %
2		Java	18.01 %	+1.2 %
3		JavaScript	9.07 %	+0.6 %
4	↑	C/C++	7.4 %	+1.1 %
5	↓	C#	7.27 %	+0.7 %
6		PHP	6.06 %	+0.0 %
7		R	4.19 %	+0.3 %
8		Objective-C	2.27 %	-1.4 %
9		Swift	1.91 %	-0.2 %
10		TypeScript	1.74 %	-0.0 %
11		Matlab	1.74 %	+0.0 %
12		Kotlin	1.71 %	+0.0 %
13		Go	1.19 %	-0.1 %
14		VBA	1.14 %	-0.0 %
15		Ruby	1.07 %	-0.0 %
16		Rust	0.98 %	-0.0 %
17	↑↑↑	Ada	0.75 %	+0.1 %
18	↑↑↑	Dart	0.72 %	+0.2 %
19	↑↑↑	Abap	0.66 %	+0.2 %
20	↓↓	Visual Basic	0.64 %	-0.0 %

Source: PYPL

Script Programming

- Scripting languages are programming languages that don't require an explicit compilation step.

    - Lua, JS, VBScript, Perl

    - Scripting is usually used to describe the rapid and flexible mode of development that Python supports

- Python is widely used without a compilation step, but the main implementation (CPython) does that by compiling to bytecode on-the-fly and then running the bytecode in a VM, and it can write that bytecode out to files (.pyc, .pyo) for use without recompiling.

- What are the differences between compilers and interpreters?

How Python runs your program?

- In Python
  - When you write a Python program, the Python interpreter reads your program and carries out the instructions it contains.
  - Compilation will generate byte code which converts into Python Virtual Machine
  - Compilation is simply a translation step, and byte code is a lower-level code.
  - But the compilation is hidden from programmer
    - If the Python process has write access on your machine, it will store the byte code of your programs in files that end with a .pyc extension (".pyc" means compiled ".py" source).
  - Why Byte code?
    - For startup speed optimization
    - In next startup simply load the .pyc files and skip the compilation step
  - What is PVM?
    - Simply a big loop that carries out byte code instructions one by one
    - Built-in Run-time engine of python
    - In simple words: PVM is a part of python system that runs your script
      - Python byte code is not binary machine code (e.g., instructions for an Intel chip). Byte code is a Python-specific representation
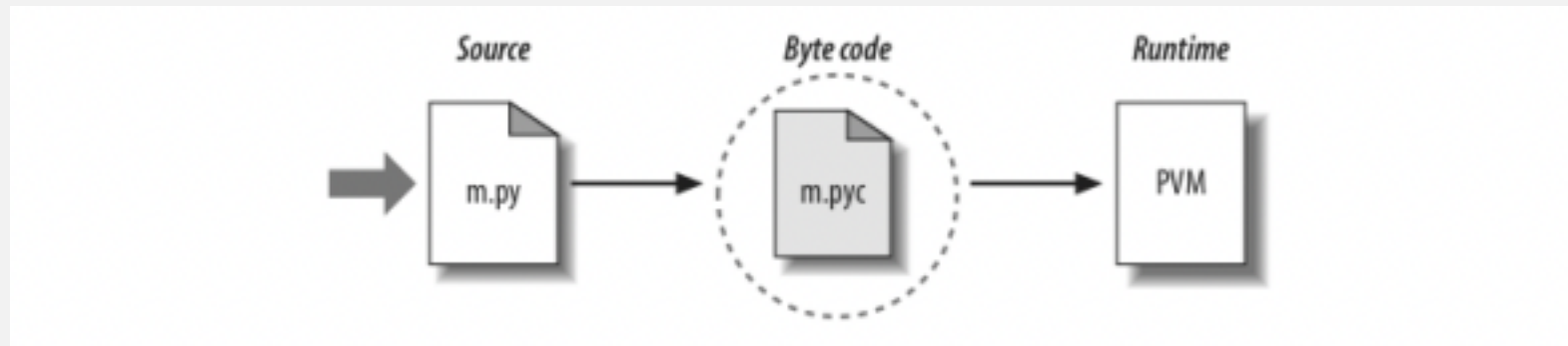
Fig: How python runs program?

Source: Learning Python, Mark Lutz

Installing Python

- Official site: https://www.python.org/downloads/

- Python (CPython) is pre-installed on most Unix systems, including Linux and OS X
  - CPython is reference implementation of the Python programming language written in C.

- 3.10 is the latest version

- Python comes with a large library of standard modules

Deciding on an IDE

There are several options for an IDE:

- IDLE or PyCharm work for most OSs, use PyCharm Pro (free for students)

- Emacs with python-mode or your favorite text editor

- Eclipse with Pydev (http://pydev.sourceforge.net/)

- You can use XCode and the Terminal

- Use VSCode and CMD to run it

Info on IDEs

- Pydev
  - Platform: GNU/Linux/macOS/Windows/Solaris
  - Official website: https://www.eclipse.org/; http://pydev.org/
  - Type: IDE
- Pycharm
  - Platform: Linux/macOS/Windows
  - Official website: https://www.jetbrains.com/pycharm/
  - Type: IDE
- Sublime Text
  - Platform: Linux/macOS/Windows
  - Official website: http://www.sublimetext.com/
  - Type: Python Text editor
- Atom/Atom-IDE
  - Platform: Linux/macOS/Windows
  - Official website: https://atom.io/
  - Type: IDE

- Visual Studio Code
  - Platform: Linux/macOS/Windows
  - Official website: https://code.visualstudio.com
  - Type: IDE

At Command Prompt

Free Python IDE	Python IDE for Mac	Python IDE for Windows
•PyDev •Visual Studio Code •Spyder •Thonny	•PyDev •Pycharm •Visual Studio Code •Spyder •Thonny	•PyDev •Pycharm •Visual Studio Code •Spyder •Thonny

Source: Hackr.io

Notebooks

- Jupyter Notebook
  - https://anaconda.org/anaconda/jupyter
- Repl.it
  - https://replit.com/G

- Google Colab
  - https://colab.research.google.com/

Variables in Python

- Variables are created when they are first assigned values.
- Variables are replaced with their values when used in expressions.
- Variables must be assigned before they can be used in expressions.
- Variables refer to objects and are never declared ahead of time

Rules for creating variables in Python

- A variable name must start with a letter or the underscore character.

- A variable name cannot start with a number.

- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).

- Variable names are case-sensitive (name, Name and NAME are three different variables).

- The reserved words(keywords) cannot be used naming the variable.

Common rules in Python

- Python is case-sensitive
- Function begin with lowercase
- Classes begins with a capital letter
- Special characters cannot be used in names

Why my code is giving
an error?
Everything looks correct

Ooops,
Indentation is
not correct

Python is sensitive to indentations

Some Examples

- Legal variable names:
    - myname = "John"
    - my_name = "John"
    - _my_nme = "John"
    - myName = "John"
    - MYNAME = "John"
    - myname2 = "John"

- Illegal variable names:
  2myname = "John"
  my-name = "John"
  my name = "John"

If..Else Statements

```
if <test1> :
    <statements1>
elif <test2> :
    <statement2>
else:
    <statement3>
```

```
if <test1> :
    <statements1>
    if <test1.2>:
        <statement1.2>
            if <test1.2.3>
                <statement1.2.3>
elif <test2> :
    <statement2>
else:
    <statement3>
```

For Loops

Syntax:

for iterator_var in sequence

        statements(s)

for n in range(x):

        print(n)

While loop

Syntax:

while expression:

      statement(s)

```python
while(n<x):
        print(n)
        n=n+2
```