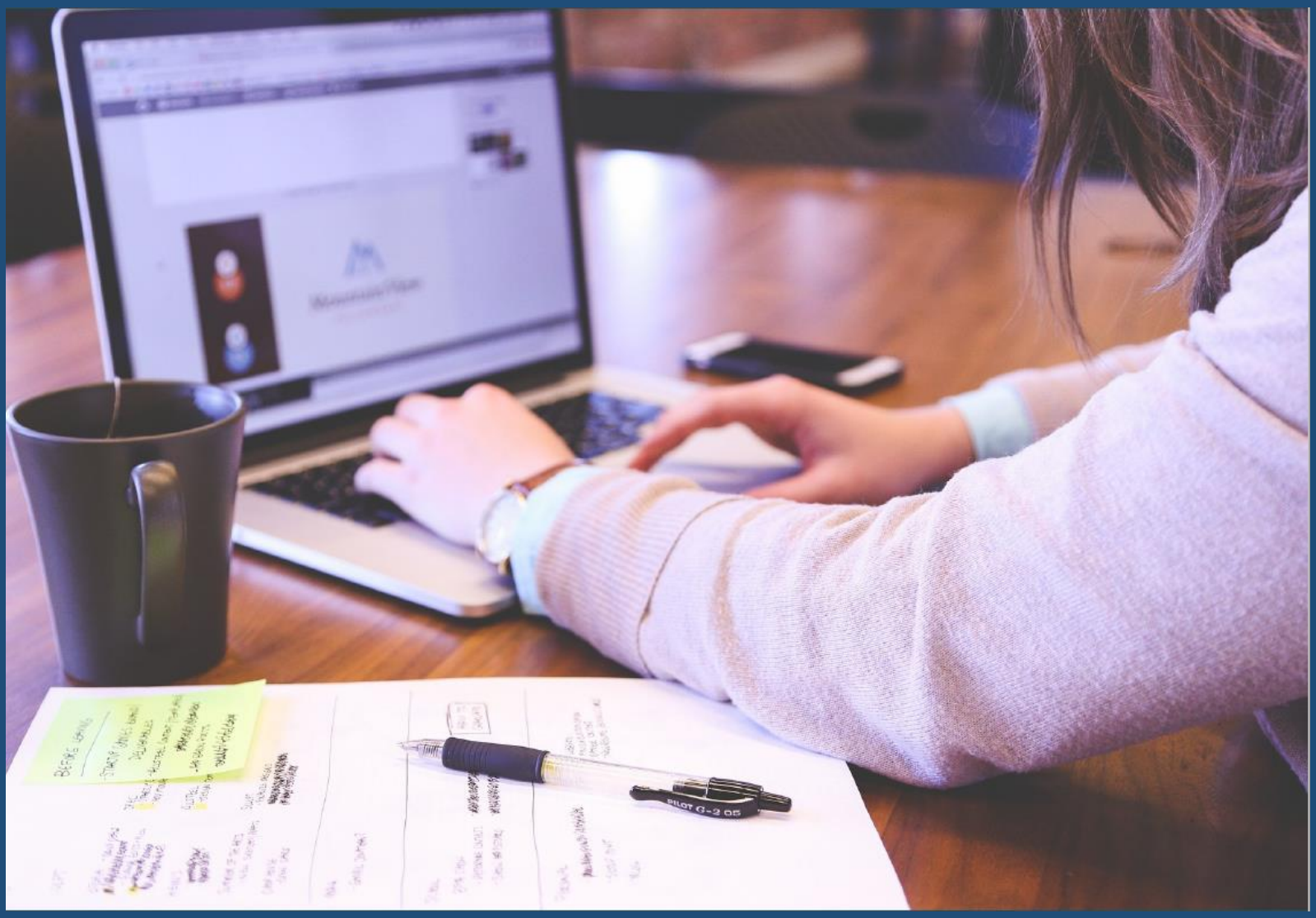
PYTHON BASICS



Ragesh Hajela

Senior Programmer, Amadeus Labs

Contact: +91-9620121987, rageshhajela@gmail.com

Training Syllabus

1. Introductory Sessions:

a. History of Python

b. Introduction

c. Starting with Python

d. Execute python script

Topic 1a: History of Python

Easy as ABC

- What do the alphabet and the programming language Python have in common? Right, both start with ABC.
- ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python.
- Python was conceptualized in the late 1980s.
- Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system.
- He experienced many frustrations with ABC and decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems.
- He created a simple virtual machine, a simple parser, and a simple runtime. He made his own version of the various ABC parts that he liked.
- He created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of

powerful data types: a hash table (or dictionary), a list, strings, and numbers.

What about the name “Python”?

- Most people think about snakes, and even the logo depicts two snakes, but the origin of the name has its root in British humour.
- During Christmas vacation in 1989, he decided to write an interpreter for the new scripting language: a descendant of ABC.
- He chose Python as a working title for a “hobby” project, being in a slightly irreverent mood during Christmas’1989 (and a big fan of Monty Python's Flying Circus).

Development Steps of Python?

- Rossum published the first version of Python code in February 1991.
- Python version 1.0 was released in January 1994.
- In October 2000, Python 2.0 was introduced.
- Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 was released in 2008.

- Python 3 is not backwards compatible with Python 2.x
- Some changes in Python 3.0:
  - Print is now a function
  - Views and iterators instead of lists
  - The division of two integers returns a float instead of an integer. "//" can be used to have the "old" behavior.
  - raw_input() in Python 2.x is changed to input() & the former is discontinued.
  - input() in Python 2.x is changed to eval(input)

Topic 1b: Introduction

Features

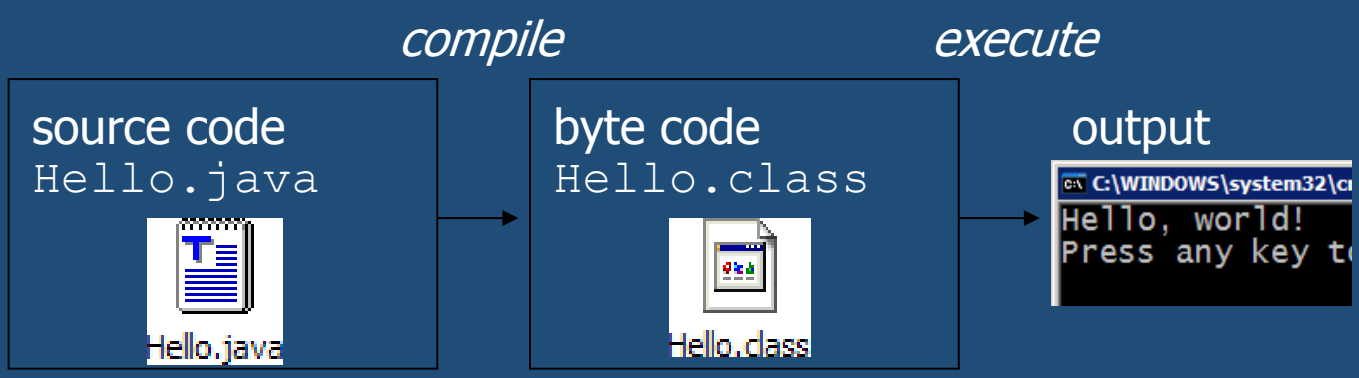
- Multi-purpose (Web, Data Science, Automation, etc.)
- Procedural and Object-Oriented
- Interpreted
- Interactive Shell
- Strongly typed and Dynamically typed
- Focus on readability and productivity
- Cross Platform (CPython, Jython, IronPython, PyPy)

Few Big Organizations using Python

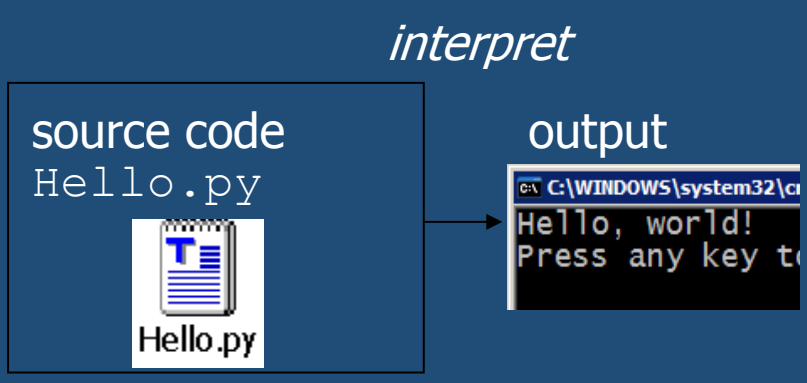
- Google
- NASA
- Netflix
- Dropbox
- YouTube
- New York University
- General Electric
- Juniper Networks
- Lego
- Nasdaq
- And the list goes on…

Compiled v/s Interpreted Language

- **Many languages require you to *compile* (translate) your program into a form that the machine understands.**



- **Python is instead directly *interpreted* into machine instructions.**



That's why Python is an interpreted language and not a compiled one. But, more insights on this topic will come in Topic 1d.

Topic 1c: Starting with Python

The Interpreter, an Interactive Shell

- The interactive shell is between the user and the operating system (e.g. Linux, Unix, Windows or others)
- The Python interpreter can be used from an interactive shell.
- The interactive shell is also interactive in the way that it stands between the commands or actions and their execution
- Python offers a comfortable command line interface with the Python shell, which is also known as the "Python interactive shell".

Setup

- We can access Python in four ways:
  - Download & install Python latest version from https://www.python.org/downloads/, and then launch IDLE. It is the interactive development environment for Python. We can access it from Windows Search Menu.
  - We can also launch Python – Command line directly after installation, instead of IDLE

- The Python interpreter can also be invoked by launching windows command prompt & typing the command "python" without any parameter followed by the "return" key at the shell prompt:

```
python
```

Python comes back with the following information

```
$ python
Python 2.7.11+ (default, Apr
2016, 14:00:29)
[GCC 5.3.1 20160413] on linu
Type "help", "copyright",
"credits" or "license" for m
information.
>>>
```

This will need to set the environment variable in Advanced System Settings.

- Finally, other way is to download a Python IDE (Interactive Development Environment) such as PyCharm, Jupiter etc. These also have facilities for debugging a program using breakpoints.

- Recommended:
  - We will use IDLE for initial classes and then move to PyCharm IDE.

Let’s try some simple commands

```
>>> hello
Traceback (most recent call
last):
  File "<stdin>", line 1, in
<module>
NameError: name 'hello' is not
defined
>>>
```

```
>>> print("Hello World")
Hello World
>>>
```

```
>>> "Hello World"
'Hello World'
>>> 3
3
>>>
```

- **How to quit the Python Shell?**
  - **Exit(), or**
  - **Quit()**

Let’s try a simple calculator

```
>>> 4.567 * 8.323 * 17
646.1893969999999
>>>
```

- Python follows the usual order of operations in mathematical expressions.
- The standard order of operations is expressed in the following enumeration:
  - exponents and roots
  - multiplication and division
  - addition and subtraction
- This means that we don't need parenthesis in the expression "3 + (2 * 4)”

```
>>> 3 + 2 * 4
11
>>>
```

- The most recent output value is automatically stored by the interpreter in a special variable with the name "_" and can be used in other expressions like any other variable

```
>>> _ * 3
33
>>>
```

- **The underscore variable is only available in the Python shell. It's NOT available in Python scripts or programs.**

Let’s try Basic Variables & Strings

```
>>> maximal = 124
>>> width = 94
>>> print(maximal - width)
30
>>>
```

```
>>> "Hello" + " " + "World"
'Hello World'
```

```
>>> ".-." * 4
'.-..-..-..-.'
>>>
```

Topic 1d: Execute python script

First Python Script

```
Select C:\Windows\System32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Users\rhajela\AppData\Local\Programs\Python\Python36-32>python first.py
My first python script!

C:\Users\rhajela\AppData\Local\Programs\Python\Python36-32>_
```

```
print("My first python script!")
```

Python Internals

- As we have studied earlier that Python language is an interpreted programming or a script language.
- The truth is: Python is both an interpreted and a compiled language. But calling Python a compiled language would be misleading.
- What is a compiler?
  - A compiler is a computer program that transforms (translates) source code of a programming language (the source language) into another computer language (the target language), mostly assembly or machine code.
  - An interpreter is a computer program that executes instructions written in a programming language. It can either

- execute the source code directly or
- translates the source code in a first step into a more efficient representation and executes this code.

- People would assume that the compiler translates the Python code into machine language. Python code is translated into intermediate code, which has to be executed by a virtual machine, known as the PVM, the Python virtual machine.
- There is even a way of translating Python programs into Java byte code for the Java Virtual Machine (JVM). This can be achieved with Jython.
- Question: Do we have to compile our Python scripts to make them faster or how can we compile them? The answer is easy: No, you don't need to do anything because "Python" is doing the thinking for you automatically.
- But, if still we want to compile, can be achieved.

```
>>> import py_compile
>>>
py_compile.compile('my_first_s
mple_script.py')
>>>
```

Or

```
python -m py_compile
my_first_simple_script.py
```

- As a result, there will be a new subdirectory "__pycache__" created, if it hasn't already existed. And also find a file "my_first_simple_script.cpython-34.pyc" in this subdirectory. This is the compiled version of our file in byte code.
- We can also compile all scripts as this:

```
monty@python:~/python$ python -
m compileall .
Listing . ...
```

- How does .pyc file generate? - If Python has write-access for the directory where the Python program resides, it will store the compiled byte code in a file that ends with a .pyc suffix. If Python has no write access, the program will work anyway. The byte code will be produced but discarded when the program exits.