Introduction to Python

BY : Reem Othman

Course Outline

1. Python History
2. Compilers Vs Interpreters
3. Python Usage
4. Python Syntax
5. Variables and Data types
6. Conditions and loops
7. Functions
8. Scopes
9. Modules
10. Introduction to OOP.

Day1 Outline

1. Python History
2. Compilers Vs Interpreters
3. Python Usage
4. Python Syntax
5. Variables and Data types
6. Conditions and loops
7. Functions

Python History

- Python is a widely-used general-purpose, high-level programming language. It was initially designed by Guido van Rossum in 1991 and developed by Python Software Foundation.

- Python 3.10.4 is the latest stable version.

"Python has been an inspiration for many other coding languages such as Ruby, Cobra, Boo, CoffeeScript ECMAScript, Groovy, Swift Go, OCaml, Julia, etc."

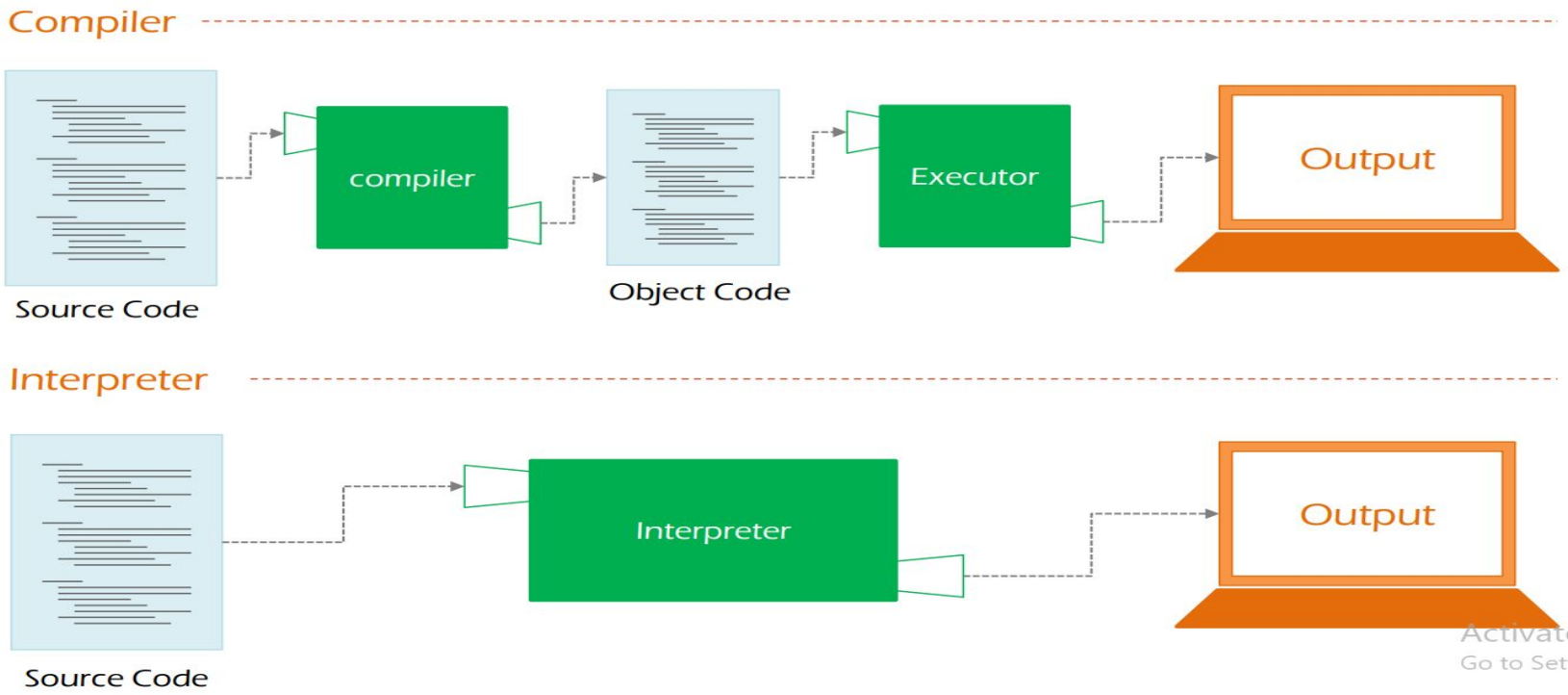
Why Python ?!

Easy To learn

Rapid Development

General Purpose Language

Compilers VS Interpreters



Compilers VS Interpreters - Cont.

	Compilers	Interpreters
Execution	A Compiler takes the entire program in one go .	An Interpreter takes a single line of code at a time
Output	The Compiler generates an intermediate machine code	The Interpreter never produces any intermediate machine code
Usage	The Compiler is best suited for the production environment.	An Interpreter is best suited for a software development environment
Languages	C , C++ , C# , Java , Scala	Python , PHP , Perl , Ruby

Python Usage

- Data analysis
- Machine learning
- Web development
- Automation or scripting
- Software testing and prototyping
- Everyday tasks

Python Syntax - identifier

A Python identifier is a name used to identify a **variable, function, class**, **module** or other object .

Starts only with:	a → z	A → Z	_	
Can't Contain :	Punctuations Characters			
Can Contain:	digits	a → z	A → Z	_

Python Syntax - reserved words

A Python identifier doesn't be one of these words

and	exec	not
assert	finally	or
break	for	pass
class	from	print
continue	global	raise
def	if	return
del	import	try
elif	in	while
else	is	with
except	lambda	yield

Python Syntax - line indentation

Vertical Alignment

```
import time                 # statement 1


def count(limit):           # statement 2
    result = 0              # statement 3
    for a in range(1, limit + 1):          # statement 4
        for b in range(a + 1, limit + 1):
            for c in range(b + 1, limit + 1):
                if c * c > a * a + b * b:
                    break

                if c * c == (a * a + b * b):
                    result += 1
    return result
```

Python Syntax - Comments

- Comment

```
# this is comment syntax in Python
```

- Paragraph

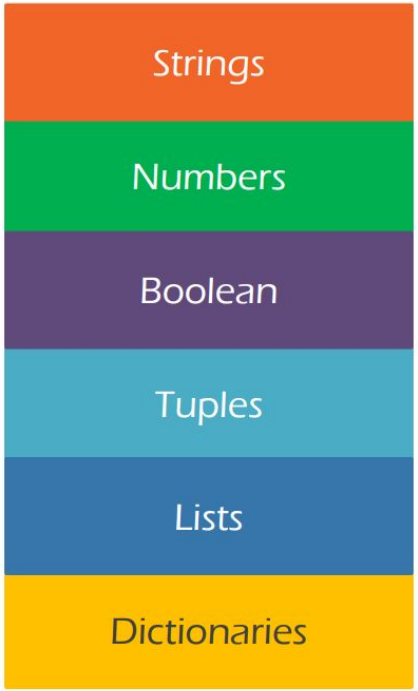
```
'''  a way to decrlare string paragraph '''
```

Variables and Data types

Variable **Identifier** = Variable **Value**

```
name = 'Reem'
age = 20
isStudent = True
age = "twenty"
```

Data Types



Data Types - Numbers Conversion

```
x = 5.16
print(int(x))
print(float(x))
print(str(x))
```

Operators

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Operators - Assignment Operator

Operator	Example	Same As
=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3
&=	x &= 3	x = x & 3
|=	x |= 3	x = x | 3
^=	x ^= 3	x = x ^ 3
>>=	x >>= 3	x = x >> 3

Operators - Comparison Operator

a <op> b

==	return True if a equals b
>=	return True if a equals or greater than b
<=	return True if a equals or lesser than b
!=	return True if a not equals b
<>	return True if a not equals b
>	return True if a greater than b
<	return True if a lesser than b

Operators - logic gates

`and`	AND Logic Gate
`or`	OR Logic Gate
`not`	Not Logic Gate

```
True and False                    #output: False
True or False                     #output: True
not False                         #output: True
not (True == 2)                   #output: True
(False == 0) and (True == 1)      #output: True
```

Strings

Declaration of string Variable

str = “this is a string “

str = ‘ this i a string too ‘

Data Structures - list

A **collection** of various data types

```
newList = [1, 'hi', True]
newList[0] #1
newList[1] #”Hi”
newList[2] #True
newList[3] #Index Error
```

Data Structures - tuples

Same as Lists but Tuples are **immutable**

```
t = (1, 'hi', True)
print(t[1]) # hi
t[1] = 4
TypeError: 'tuple' object does not support item assignment
```

Data Structures - Dictionaries

A **key: value** comma separated elements Data Structure

```
d = {'name': 'Reem', 'track': 'Data'}
d['name']
# Reem
d['name'] = 'Ali'
# {name: "Ali", track: "Data"}
```

Conditions

```
x = 3

if (x == 2):
    print('Two')
elif (x == 3):
    print('Three')
else:
    print('others')
```

Loops - for

```
languages = ['JavaScript', 'Python', 'Java']
for l in languages:
    print(l)

    #Javascript
    #Python
    #Java
```

Loops - While

```
dayCount = 0
while dayCount < 4:
    print('We are learning Python')
    dayCount += 1
```

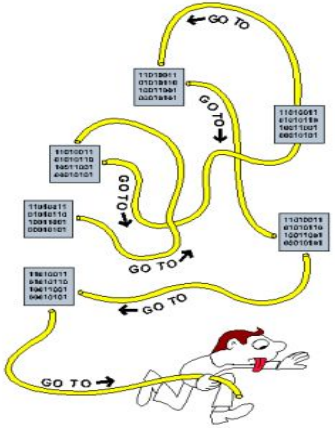
Loops - Break Statement

```
i = 1
while i < 6:
  if i == 3:
    break
  print(i)
  i += 1

# 1
# 2
```

Loops -Continue Statement

```
i = 0
while i < 6:
  if i == 3:
    continue
  i += 1
  print(i)

# 1
# 2
# 4
# 5
# 6
```

Python Levels



Speghatti Level

+ Functions

Procedural Level

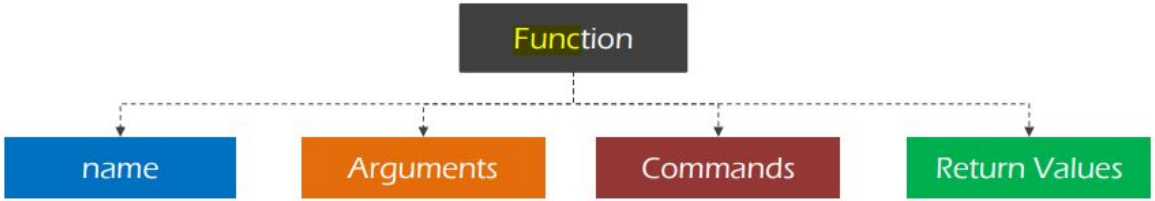
+ Modules

Modular Level

Object Oriented Level

Function Declaration

```
def fnName:
    pass
```




```
def measureTemp ( temp ):
    if temp < 37:
        return "Too Cold"
    elif temp > 37:
        return "Too Hot"
    return "Normal"
```

```
measureTemp(37)
# "Normal"
```

Functions - default argument

```
def doSum(x, y = 2,z = 3):
    sum = x + y + z
    print(sum)
```

Calling It

```
doSum(2)          # output: 7
doSum(2,4)        # output: 9
doSum(2,4,10)     # output: 16
```

Functions - arguments

```
def doSum(*args):
    sum = 0
    for i in args:
        sum += i;
    print(sum)
```

Calling It

```
doSum(2,6)          # output: 8
doSum(2,4,5,15)     # output: 26
```

Functions - keywords

```
def doSum(**kwargs):
    for k in kwargs:
        print(kwargs[k])
```

Calling It

```
doSum(x = 2, y = 26)    # output: 2
                                  26
```

Common Used Functions - Input

- Input

```
input(prompt_message)
```

Example

```
name = input("What's your Name? ");
print(name);
```

Common Used Functions - Range

- Range

```
range([start,] end[, step])
```

Examples

```
range(5)            [0,1,2,3,4]
range(0,5,1)        [0,1,2,3,4]
range(1,10,2)       [1,3,5,7,9]
```

```
for i in range(10):
    print(i)
```

0	1	2	3	4	5	6	7	8	9

Practical time

Problems

- Python Program to add 2 numbers
- Python Code that swap two numbers
- Python Program to check Prime Numbers (2, 3, 4, 5, 7).
- Write Program which has an input of a string from user then it will return the same string reversed .