



# Object Oriented Programming using Python (I)

## Lecture(2)

### Programming Paradigm

**Lecturer: Ahmed Eskander Mezher**  
***University of Information Technology and Communications***  
***College of Business Informatics***

# *What is Python...?*

- Python is a general purpose programming language that is often applied in scripting roles.
- So, Python is programming language as well as scripting language.
- Python is also called as Interpreted language

## **Scripting**

- **a script is interpreted**
- A "script" is code written in a scripting language. A scripting language is nothing but **a type of programming language in which we can write code to control another software application.**

# *Why do people use Python...?*

The following primary factors cited by Python users seem to be these:

- **Python is object-oriented**

Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance.

- **Indentation**

Indentation is one of the greatest feature in Python.

- **It's free (open source)**

Downloading and installing Python is free and easy  
Source code is easily accessible

# *Why do people use Python...?*

## □ It's mixable

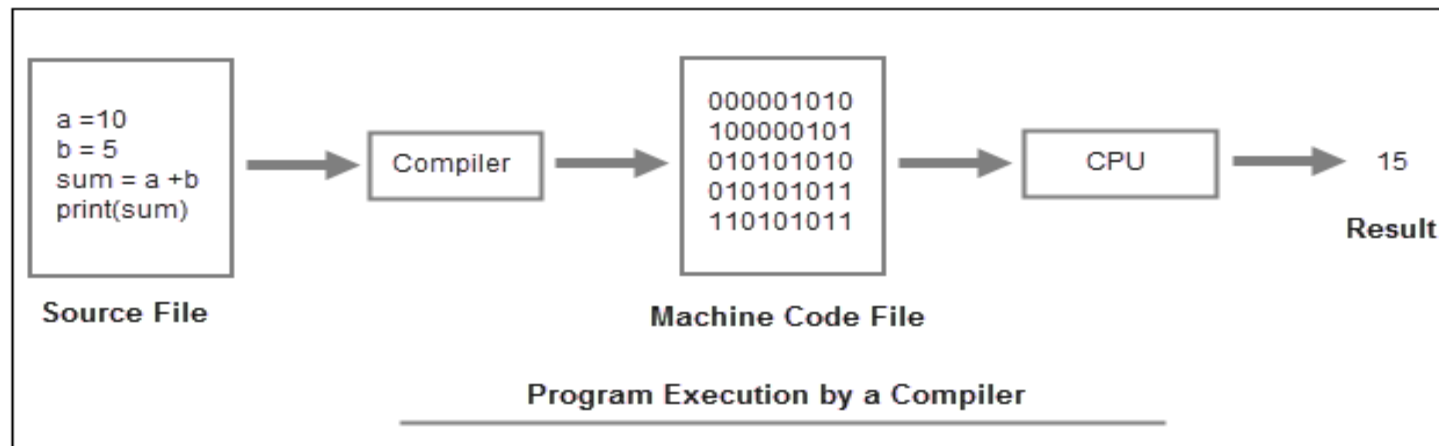
- Python can be linked to components written in other languages easily
- Linking to fast, compiled code is useful to computationally intensive problems
- - Python/C integration is quite common

## □ It's easy to use

- No intermediate compile and link steps as in C/ C++
- Python programs are compiled automatically to an intermediate form called *bytecode*, which the interpreter then reads
- This gives Python the development speed of an interpreter without the performance loss inherent in purely interpreted languages

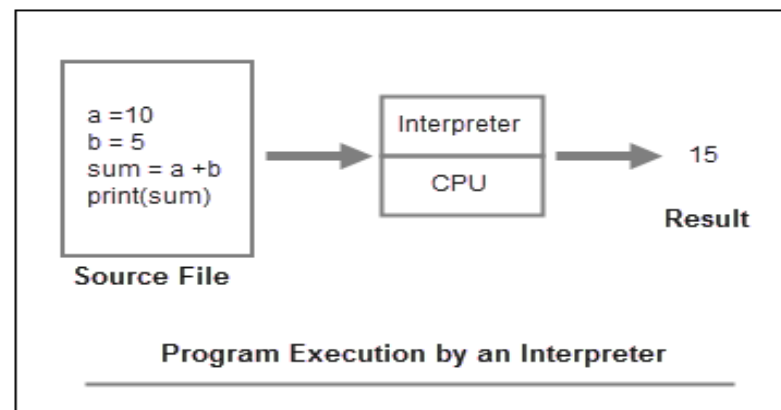
# Compiler

A compiler translates the entire source code into machine language in one go, the machine language is then executed.



## Interpreter

An interpreter, on the other hand, translates high-level language into machine language line by line, which it then executes. Python Interpreter starts at the top of the file, translates the first line into machine language and then executes it. This process keeps repeating until the end of the file is reached.




# Boolean Expressions

Arithmetic expressions evaluate to numeric values; a Boolean expression, have only one of two possible values: false or true.

**Listing 4.1: boolvars.py**

```
# Assign some Boolean variables
a = True
b = False
print('a =', a, ' b =', b)
# Reassign a
a = False;
print('a =', a, ' b =', b)
```



```
a = True  b = False
a = False b = False
```

Expression	Meaning
$x == y$	True if $x = y$ (mathematical equality, not assignment); otherwise, false
$x < y$	True if $x < y$ ; otherwise, false
$x \leq y$	True if $x \leq y$ ; otherwise, false
$x > y$	True if $x > y$ ; otherwise, false
$x \geq y$	True if $x \geq y$ ; otherwise, false
$x \neq y$	True if $x \neq y$ ; otherwise, false




# If statement

A program sometimes may have to make choices. These choices can execute different code depending on certain condition.

To validate if input is equal use the == operator (not =).

```
age = eval(input("enter ur age? "))
if age < 15:
    print("Your are child.")
else:
    print("Your are young.")
|
```



```
>>>
enter ur age? 12
Your are child.
>>> ===== RESTART =====
>>>
enter ur age? 25
Your are young.
>>> |
```

# If else statement

```
# Get two integers from the user
dividend, divisor = eval(input('Please enter two numbers to divide: '))
# If possible, divide them and report the result
if divisor != 0:
    print(dividend, '/', divisor, "=", dividend/divisor)
else:
    print('Division by zero is not allowed')
```

## The result

Please enter two numbers to divide: 32, 8  
32 / 8 = 4.0

Please enter two integers to divide: 32, 0  
Division by zero is not allowed



# Compound Boolean Expressions

Simple Boolean expressions, each involving one relational operator, can be combined into more complex Boolean expressions using the logical operators and, or, and not. A combination of two or more Boolean expressions using logical operators is called a compound Boolean expression.

$e_1$	$e_2$	$e_1$ and $e_2$	$e_1$ or $e_2$	not $e_1$
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False

! Logical operators— $e_1$  and  $e_2$  are Boolean expressions

Python allows an expression like:

**$x \leq y$  and  $y \leq z$**

# Compound Boolean Expressions

```
value = eval(input("Please enter an integer value in the range 0...10: "))
if value >= 0 and value <= 10: # Only one, more complicated check
    print("In range")
print("Done")
```

```
value = eval(input("Please enter an integer value in the range 0...10: "))
if value >= 0:      # First check
    if value <= 10: # Second check
        print(value, "is in range")
    else:
        print(value, "is too large")
else:
    print(value, "is too small")
print("Done")
```

# Iteration

Iteration repeats the execution of a sequence of code

## ■ The while Statement

```
count = 1      # Initialize counter
while count <= 5: # Should we continue?
    print(count) # Display counter, then
    count += 1  # Increment counter
```



```
1
2
3
4
5
>>>
```

**print(count)**

**count += 1**



are executed five times. After each redisplay of the variable count, the program increments it by one. Eventually (after five iterations), the condition count <= 5 will no longer be true, and the block is no longer executed.

# Iteration

## ■ For Statement

Python provides a more convenient way to express a definite loop. The for statement iterates over a range of values.

```
n = 1
while n <= 10:
    print(n)
    n += 1
```

The while loop can be rewritten

```
for n in range(1, 11):
    print(n)
```

*range ( begin, end, step )*

where

- begin is the first value in the range; if omitted, the default value is 0
- end is one past the last value in the range; the end value may not be omitted
- step is the amount to increment or decrement; if the change parameter is omitted, it defaults to 1 (counts up by ones)begin, end, and step must all be integer values; floating-point values and other types are not allowed.

# examples

```
for n in range(21, 0, -3):  
    print(n, ' ', end='')
```

It prints

```
21 18 15 12 9 6 3
```

The following code computes and prints the sum of all the positive integers less than 100:

```
sum = 0    # Initialize sum
```

```
for i in range(1, 100):
```

```
    sum += i
```

```
print(sum)
```

# The break statement

In the following example while loop breaks when the count value is 5. The print statement after the while loop displays the value of num\_sum (i.e. 0+1+2+3+4).

```
1 num_sum = 0
2 count = 0
3 while(count<10):
4     num_sum = num_sum + count
5     count = count + 1
6     if count== 5:
7         break
8 print("Sum of first ",count,"integers is: ", num_sum)
```

Output:

```
Sum of first 5 integers is : 10
```

# The continue Statement

## continue statement

---

The continue statement is used in a while or for loop to take the control to the top of the loop without executing the rest statements inside the loop. Here is a simple example.

```
1 for x in range(7):
2     if (x == 3 or x==6):
3         continue
4     print(x)
```

Output:

```
0
1
2
4
5
```

In the above example, the for loop prints all the numbers from 0 to 6 except 3 and 6 as the continue statement returns the control of the loop to the top



# Function

- A **function** is a unit of reusable code
- Code is made more reusable by packaging it in functions.
- we will see how to write our own reusable functions and examine some of functions available in the Python standard library.
- Python provides a collection of standard code stored in libraries called **modules**.
- Programmers can use parts of this library code within their own code to build their programs.

# Built-in Functions

Python has several functions that are readily available for use. These functions are called built-in functions. In this reference page, you will find all the built-in functions in Python.

## example

This program use **sqrt** function to find square root of number and **math** library that contain sqrt function

```
from math import sqrt

# Get value from the user
num = eval(input("Enter number: "))

# Compute the square root
root = sqrt(num)

# Report result
print("Square root of", num, "=", root)
```

# Function

```
from math import sqrt
```

- makes the sqrt function available for use in the program.
- Math is **Module** a collection of Python code that can be used in other programs .
- math module has many other mathematical functions. These include logarithmic, hyperbolic, and other mathematical functions. When calling a function, the function's name is followed by parentheses that contain the information to pass to the function so it can perform its task.

```
sqrt(num)
```

# Math functions module

mathfunctions Module	
sqrt	Computes the square root of a number: $\text{sqrt}(x) = \sqrt{x}$
exp	Computes $e$ raised a power: $\text{exp}(x) = e^x$
log	Computes the natural logarithm of a number: $\text{log}(x) = \log_e x = \ln x$
log10	Computes the common logarithm of a number: $\text{log}(x) = \log_{10} x$
cos	Computes the cosine of a value specified in radians: $\text{cos}(x) = \cos x$ ; other trigonometric functions include sine, tangent, arc cosine, arc sine, arc tangent, hyperbolic cosine, hyperbolic sine, and hyperbolic tangent
pow	Raises one number to a power of another: $\text{pow}(x,y) = x^y$
degrees	Converts a value in radians to degrees: $\text{degrees}(x) = \frac{\pi}{180}x$
radians	Converts a value in degrees to radians: $\text{radians}(x) = \frac{180}{\pi}x$
fabs	Computes the absolute value of a number: $\text{fabs}(x) =  x $

- The math package also defines the values( pi )and (e).

# examples

```
# This program shows the various ways the
# sqrt function can be used.

from math import sqrt

x = 16
# Pass a literal value and display the result
print(sqrt(16.0))
# Pass a variable and display the result
print(sqrt(x))
# Pass an expression
print(sqrt(2 * x - 5))
# Assign result to variable
y = sqrt(x)
print(y)
# Use result in an expression
y = 2 * sqrt(x + 16) - 4
print(y)
# Use result as argument to a function call
y = sqrt(sqrt(256.0))
print(y)
print(sqrt(int('45')))
```

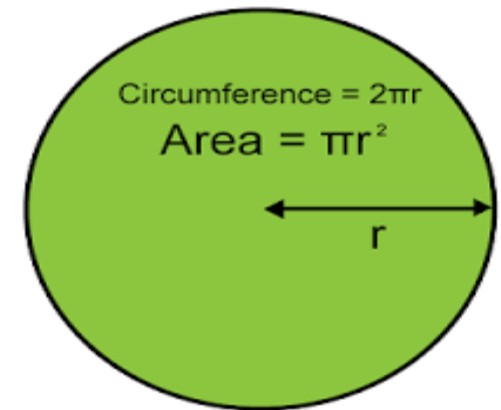
■ Also We can use math functions as following:

**y = math.sqrt(x)**

# examples

This program to find circumference and area of circle

```
from math import *  
  
r=eval(input("enter raduis of circle  
x=2*r*pi  
y=pow(r,2)*pi  
print("circumference of a circle",x)  
  
print("area of a circle",y)  
  
print(pi)
```



# Random functions

Some applications require behavior that appears random. Random numbers are useful particularly in games and simulation.

## randomfunctions Module

random

Returns a pseudorandom floating-point number  $x$  in the range  $0 \leq x < 1$

randrange

Returns a pseudorandom integer value within a specified range.

```
from random import randrange

for i in range(0, 20):
    print(randrange(1, 100), end=' ')
print()
```

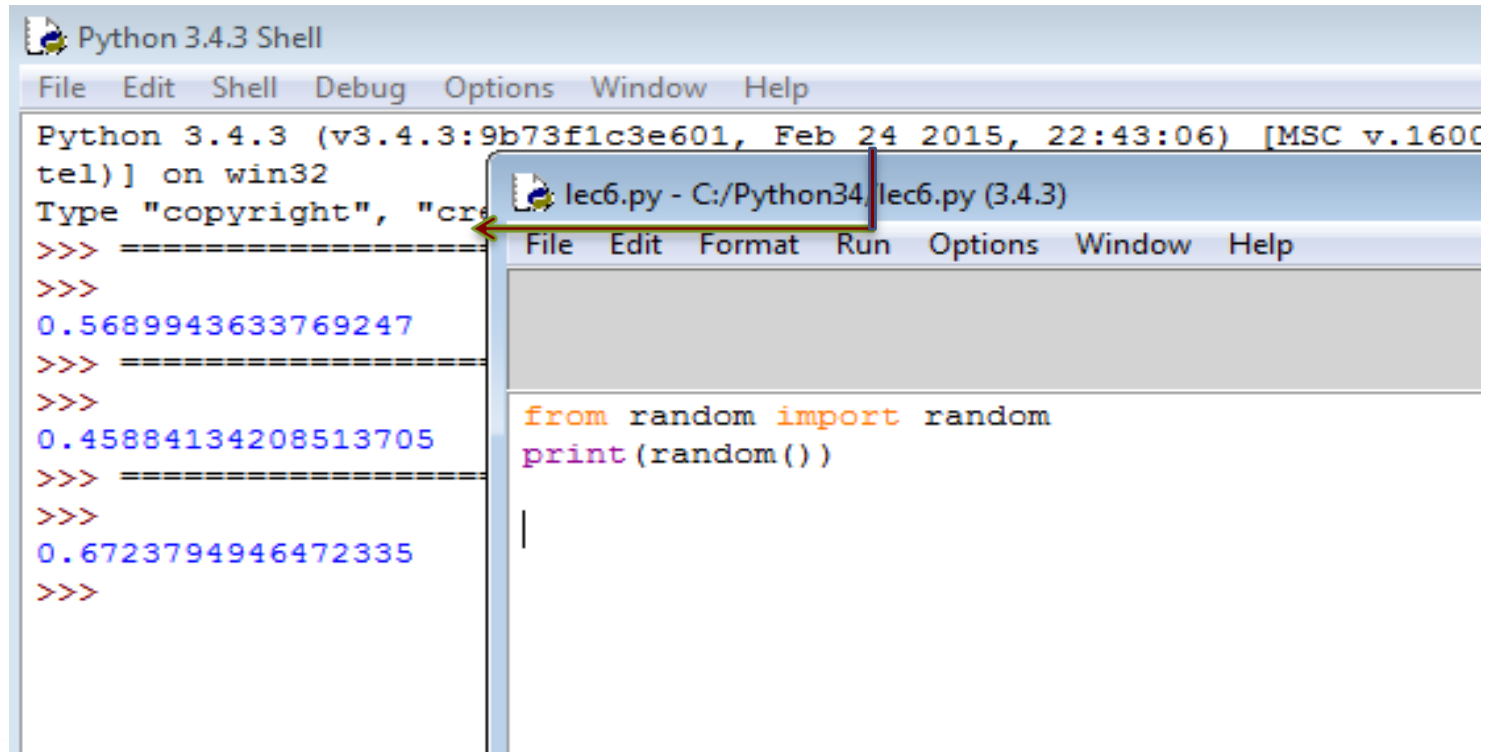
run

```
>>>
96 6 91 42 39 18 32 41 47 86 15 94 27 13 54 32 27 71 36 94
>>> |
```



# Random function

Some functions do not accept any parameters; for example, the function to generate a pseudorandom floating-point number, `random` function requires no arguments:



The image shows two overlapping windows from a Python 3.4.3 environment. The background window is the 'Python 3.4.3 Shell' with a menu bar (File, Edit, Shell, Debug, Options, Window, Help). It displays the Python version and build information, followed by a prompt 'Type "copyright", "credits" or "help()" and then three calls to the `random` function, each returning a different floating-point number: `0.5689943633769247`, `0.45884134208513705`, and `0.6723794946472335`. The foreground window is a script editor titled 'lec6.py - C:/Python34/lec6.py (3.4.3)' with a menu bar (File, Edit, Format, Run, Options, Window, Help). It contains the code `from random import random` and `print(random())`, with a cursor on the line following.

```
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600  
tel)] on win32  
Type "copyright", "credits" or "help()">>> =====  
>>>  
0.5689943633769247  
>>> =====  
>>>  
0.45884134208513705  
>>> =====  
>>>  
0.6723794946472335  
>>>  
  
lec6.py - C:/Python34/lec6.py (3.4.3)  
File Edit Format Run Options Window Help  
  
from random import random  
print(random())  
|
```

# Example Max () function

- The max() function returns the largest of the input values.

```
# find largest item in the string  
print(max("abcDEF"))
```

```
# find largest item in the list  
print(max([2, 1, 4, 3]))
```

- <https://thepythonguru.com/python-builtin-functions/>