CORVIT
Reaching end to end

Introducing Python

Lecture# 2

by

Umair bin Mansoor

Network Programming Planner

CORVIT
Reaching end to end

python™

OBJECTIVES

After this session, students will be able to:

- **To write programs that perform simple computations**
- **To obtain input from a program's user by using the `input` function**
- **To use identifiers to name elements such as variables and functions**
- **To assign data to variables**
- **To perform simultaneous assignment**
- **To define named constants**
- **To use the operators +, −, *, /, //, % and ****
- **To write and evaluate numeric expressions**
- **To use augmented assignment operators to simplify coding**
- **To perform numeric type-conversion with the `int` and `round` functions**
- **To obtain the current system time and date by importing `time` and `datetime` modules**

CORVIT
Reaching end to end

python™

WRITING A SIMPLE PROGRAM

- **Algorithm**
  - **An algorithm describes how a problem is solved by listing the actions that need to be taken and the order of their execution.**
  - **Algorithms can help the programmer plan a program before writing it in a programming language.**
  - **Algorithms can be described in natural languages or in pseudo code.**
  - **The algorithm for calculating the area of a circle can be described as follows:**
    1. **Get the circle's radius from the user**
    2. **Compute the area by applying the following formula**

       Area = radius x radius x π
    3. **Display the result**
  - ***ComputeArea.py* contains the code for the above algorithm.**

WRITING A SIMPLE PROGRAM

ComputeArea(2_1).py

```
from math import pi
radius = float(input("Input the radius of the Circle: "))
Area = pi * radius * radius
print("The Area of the circle is ", Area)
```

ComputeAverage(2_2).py

```
number1 = eval(input("Enter the first number: "))
number2 = eval(input("Enter the second number: "))
number3 = eval(input("Enter the third number: "))
average = (number1 + number2 + number3) / 3
print("The average of", number1,",", number2,",", number3, "is",\
average)
```

IDENTIFIERS

- In the previous program , `number1`, `number2`, `number3`, `input` are all names known as identifiers.
  - An identifier is a sequence of characters that consists of letters, digits, and underscores (_).
  - An identifier must start with a letter or an underscore and not with a number.
  - A user defined identifier/variable cannot be a Python keyword.
  - An identifier can be of any length.
- For example, `Area`, `radius`, and `number1` are legal identifiers, where as `2A` and `d+4` are not because they do not follow the rules.
- Because Python is case sensitive, `area`, `Area` and `AREA` are all different identifiers.

CORVIT
Reaching end to end

python™

VARIABLES, ASSIGNMENT STATEMENTS AND EXPRESSIONS

- In Python, variables are actually objects and are treated as references to the memory locations in a computer.
- In the previous example, `Area`, `radius` are variables. They are called variables because they can reference different memory locations.
- Assignment statements sets a variable to reference a variable/constant/expression.
  - `radius = 20`
  - `Area = radius * radius * pi` ⟹ *(expression made of variables)*
  - `Area = 20**2 * 3.14159` ⟹ *(expression made of constants)*
- Valid assignments:
  - `x = x + 1` ⟹ *adds 1 to the variable x and then updates x*
  - `i = j = k = 1` ⟹ *assigns 1 to the variables i, j, k simultaneously*

NUMERIC DATA TYPES AND OPERATORS

- The information stored in a computer is generally referred to as *data*.
- There are two types of numeric data: integers and real number
  - Integer type (*int*) represents whole numbers
  - Real type (*float*) represents numbers with fractional part
- Python interpreter differentiates between different types of numbers itself.
- There are several other data types/container classes which will be discussed later.
- Operators operates on operands. There are a few classes of operators:
  - Numeric/Arithmetic operators
  - Assignment/Augmented assignment operators
  - Relational operators
  - Logical operators
  - Unitary operators

Binary operators

NUMERIC/ARITHMETIC OPERATORS

TABLE 2.1 Numeric Operators

Name	*Meaning*	*Example*	*Result*
+	Addition	34 + 1	35
-	Subtraction	34.0 - 0.1	33.9
*	Multiplication	300 * 30	9000
/	Float Division	1 / 2	0.5
//	Integer Division	1 // 2	0
**	Exponentiation	4 ** 0.5	2.0
%	Remainder	20 % 3	2

NUMERIC/ARITHMETIC OPERATORS

TABLE 2.2 Augmented Assignment Operators

Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Float division assignment	`i /= 8`	`i = i / 8`
`//=`	Integer division assignment	`i //= 8`	`i = i // 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`
`**=`	Exponent assignment	`i **= 8`	`i = i ** 8`

NUMERIC/ARITHMETIC OPERATORS EXAMPLE

- ArithmeticOperators(2_3).py

```
x = 15
y = 4
print('x + y =', x+y)     #Addition operator
print('x - y =', x-y)     #Subtraction operator
print('x * y =', x*y)     #Multiplication operator
print('x / y =', x/y)     #Division operator
print('x // y =', x//y)   #Integer Division operator
print('x % y =', x%y)     #Remainder operator
print('x ** y =', x**y)   #Power operator
```

RELATIONAL OPERATORS

- relationalOperators(2_4).py

```
x = 10
y = 12
print('x > y  is ', x>y)     # greater than
print('x < y  is ', x<y)     # less than
print('x == y is ', x==y)    # equals to
print('x != y is ', x!=y)    # not equal to
print('x >= y is ', x>=y)    # greater than or equal to
print('x <= y is ', x<=y)    # less than or equal to
```

LOGICAL OPERATORS

- logicalOperators(2_5).py

```
1. x = True
2. y = False

3. print('x and y is ', x and y)  # x and y is False
4. print('x or y is ', x or y)    # x or y is True
5. print('not x is ', not x)      # not x is False
```

TYPE CONVERSION AND ROUNDING

- **If an integer and a float variable is added and assigned to another variable, Python converts it to the flot data type.**
- **int() function converts a float or a string version of integer to an integer value:**
- **Similarly, a float() function converts an integer or a string versión of float to a float value.**
  - `int(3.5)  # outputs value 3 by truncating`
  - `int("35")  # outputs value 35`
  - `int("35.5") # outputs ValueError`

- **Round() function rounds of a floating point value to the nearest integer value.**
  - `round(3.5)# outputs value 4`
  - `round(3.4)# outputs value 3`

CURRENT TIME AND DATE

- **Time module**
  - **The time module helps to find the GMT time in seconds (GMT starts from 1970)**

```
1. import time
2. print(time.time())   # outputs 1575130330.9903283
```

seconds **milliseconds**

- **Datetime module**
  - **The datetime module returns the date as well as the current time in hour mins and seconds format.**

```
1. import datetime
2. print(datetime.datetime.now()) #2019-11-30 21:19:48.334138
```

PYTHON programming exercises

2.1 (*Convert Celsius to Fahrenheit*) Write a program that reads a Celsius degree from the console and converts it to Fahrenheit and displays the result. The formula for the conversion is as follows:

```
fahrenheit = (9 / 5) * celsius + 32
```

Here is a sample run of the program:

```
Enter a degree in Celsius: 43 ↵Enter
43 Celsius is 109.4 Fahrenheit
```

PYTHON programming exercises

2.3 (*Convert feet into meters*) Write a program that reads a number in feet, converts it to meters, and displays the result. One foot is 0.305 meters. Here is a sample run:

```
Enter a value for feet: 16.5 ↵Enter
16.5 feet is 5.0325 meters
```

2.4 (*Convert pounds into kilograms*) Write a program that converts pounds into kilograms. The program prompts the user to enter a value in pounds, converts it to kilograms, and displays the result. One pound is 0.454 kilograms. Here is a sample run:

```
Enter a value in pounds: 55.5 ↵Enter
55.5 pounds is 25.197 kilograms
```

PYTHON programming exercises

**2.6 (*Sum the digits in an integer*) Write a program that reads an integer between 0 and 1000 and adds all the digits in the integer. For example, if an integer is 932, the sum of all its digits is 14. (Hint: Use the % operator to extract digits, and use the // operator to remove the extracted digit. For instance, 932 % 10 = 2 and 932 // 10 = 93.) Here is a sample run:

```
Enter a number between 0 and 1000: 999 ↵Enter
The sum of the digits is 27
```

Questions & Answers