Python Programming

High performance. Delivered.

Module 4 – Language Fundamentals - Basic Syntax , Data Types , Operators.

accenture

Strategy | Digital | Technology | Operations

Agenda

- Basic Syntax of Python Language
- Writing and executing the first Python Script
- Input and Output to Console
- Comments in Python
- Variables in Python
- Datatypes in Python   - Numbers and Strings
- Reserved Words in Python
- Types of Operators in Python

Module Objectives

At the end of this module, you will be able to understand the fundamentals of Python Language like

- Basic Syntax of Python Language
- Writing and executing the first Python Script
- Input and Output to Console
- Comments in Python
- Variables in Python
- Datatypes in Python - Numbers and Strings
- Reserved Words in Python
- Types of Operators in Python

Executing the first Python Program

Open the editor and type the following statements

```
import os
os.system('cls')
print("Hello World")
print("Welcome to Accenture....")
```

Save the file as “Hello.py”

open the terminal window and type

python Hello.py

and press enter

Output:

```
Hello World
Welcome to Accenture
```

Printing Messages in Python

- The Print statement –
  - prints the given message/ value to the console

```
import os
os.system('cls')
print ("=========== Welcome to Python World....==============")
print("Hello World")
print("Hello World !!!"+" Welcome to Accenture....")
print("\n\n\n")
print("================== Good Day....=====================")
```

```
OutPut:
=========== Welcome to Python World....==============
Hello World
Hello World !!! Welcome to Accenture....


================== Good Day....=====================
```

Keyboard input in Python

- The Input Function: accepts values from the keyboard

```
name = input("Enter Your Name: ")
print ("Your Name is = "+name)
age = input("Enter Your Age : ")
print("So, you are already " + age + " years old, " + name + "!")
```

```
OutPut:
Enter Your Name:  Accenture
Your Name is = Accenture
Enter Your Age :  16
So, you are already 16 years old , Accenture !
```

Use of Semicolon in Python Programs

- Using semicolons is not mandatory in a program.
- It is used for separating two statements

```
print("Hello World...") print("Welcome to Accenture...")
```

The above code would produce an error – SyntaxError – Invalid syntax

```
print("Hello World...") ; print("Welcome to Accenture...")
```

The above code would produce the Output

```
Hello World…
Welcome to Accenture…
```

Reserved/Keywords in Python

- Python has 33 keywords.
- Keywords should not be used to define a variable name, function name or any other identifier.
- Python, keywords are case sensitive.
- All the keywords except True, False and None are in lowercase and they must be written as it is.

Keywords in Python				
False	class	finally	is	return
None	continue	for	lambda	try
True	def	from	nonlocal	while
and	del	global	not	with
as	elif	if	or	yield
assert	else	import	pass	
break	except	in	raise	

Comments in Python

Single Line Comment

Single-line comments begin with the hash character ("#") and are terminated by the end of line.

Eg:

```
# This is a comment in Python
```

Multiline Comments

Comments spanning more than one line are achieved by inserting a multiple lines string Between two “”” (Three double quotation marks

Eg:

```
""""""
This is an example of a multiline
comment that spans multiple lines
======================
""""""
```

Variables and Data Types

- Python is completely object oriented, and not "statically typed".
- No need to declare variables before using them, or declare their type.
- Every variable in Python is an object.

```
name = input("Enter Your Name: ")
print ("Your Name is = "+name)
age = input("Enter Your Age? ")
print("So, you are already " + age + " years old, " + name + "!")
```

Here, the variables (name , age) are not declared but they are used.

Rules for identifiers in Python

Identifier is the name given to entities like class, functions, variables etc.

- Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (_).
- An identifier cannot start with a digit. 1variable is invalid, but variable1 is perfectly fine.
- Keywords cannot be used as identifiers.

Valid Identifiers
myClass,
var_1
_myvar
Myvar_

Invalid Identifiers
My Class,
1var
Var*v

Data Types in Python

Every value in Python has a datatype. Everything is an object in Python programming,
datatypes are actually classes and variables are instance (object) of these classes.
There are various datatypes in Python.

Some of the important types are listed below.

Numbers
Strings
List
Tuple
Set
Dictionary

*Note: **Numbers** and **Strings** are briefly discussed in this module, and the rest of the data types are discussed in detail in the succeeding modules.*

Numbers in Python

- Python supports two types of numbers - integers and floating point numbers
- To define an integer and float , use the following syntax.

```
import os
os.system('cls')
myint = 7
myfloat1 = 7.65
myfloat2 = float(7.9265)
print("MyInt = "+str(myint))
print("MyFloat1 = "+str(myfloat1))
print("MyFloat2 = "+str(myfloat2))
```

```
OutPut:
MyInt = 7
MyFloat1 = 7.65
MyFloat2 = 7.9265
```

Complex Numbers

Python also supports Complex numbers

To use complex numbers use the following Syntax

```
import os
os.system('cls')
c1 = complex(2,3)
print("Complex no C1 = "+str(c1))
print("Real Part      = "+str(c1.real))
print("Imaginary Part= "+str(c1.imag))
print("Conjugate = "+str(c1.conjugate()))
```

```
OutPut
=======
Complex no C1 = (2+3j)
Real Part      = 2.0
Imaginary Part= 3.0
Conjugate = (2-3j)
```

Strings

Strings are defined either with a single quote or a double quotes.

```
mystring = 'hello'
mystring = "Hello"
```

The difference between the two is that using double quotes makes it easy to include apostrophes

```
mystring = "Don't worry about apostrophes"
```

Strings Contd…

Mixing operators between numbers and strings is not supported

```
x=11
y="Hello"
xy = x + y          # This will not work
print (xy)
```

The following error is displayed by the compiler

```
TypeError: unsupported Operand type(s) for +: 'int' and 'str'
```

Operators in Python

Python has the following types of operators

- Arithmetic operators
- Boolean Operators
- Comparison Operators
- Logical Operators
- Bitwise Operators
- Assignment operators
- Special operators (Identity operators and Membership operators)

Arithmetic operators

Operator	Meaning	Example
+	Add two operands or unary plus	x + y +2
-	Subtract right operand from the left or unary minus	x - y -2
*	Multiply two operands	x * y
/	Divide left operand by the right one (always results into float)	x / y
%	Modulus - remainder of the division of left operand by the right	x % y (remainder of x/y)
//	Floor division - division that results into whole number adjusted to the left in the number line	x // y
**	Exponent - left operand raised to the power of right	x**y (x to the power y)

Airthmetic Operators - Example

```
x = 15
y = 4
print('x + y = ',x+y)
print('x - y = ',x-y)
print('x * y = ',x*y)
print('x / y = ',x/y)
print('x // y = ',x//y)
print('x ** y = ',x**y)
```

Output

```
x + y =  19
x - y =  11
x * y =  60
x / y =  3.75
x // y =  3
x ** y =  50625
```

Boolean Operators (Logical Operators)

Python has the following three Boolean operators

- and,
- or,
- not

Operation	Result
x or y	if x is false, then y, else x
x and y	if x is false, then x, else y
not x	if x is false, then True, else False

Notes:
or - This is a short-circuit operator, so it only evaluates the second argument if the first one is False.
and - This is a short-circuit operator, so it only evaluates the second argument if the first one is True.
not - not has a lower priority than non-Boolean operators, so not a == b is interpreted as not (a == b),
and a == not b is a syntax error.

Boolean Operators (Logical Operators) - Example

```
x = True
y = False
print('x and y is',x and y)
print('x or y is',x or y)
print('not x is',not x)
```

Output:

```
x and y is False
x or y is True
not x is False
```

Comparison Operators (Relational Operators)

Operator	Meaning	Example
>	Greater that - True if left operand is greater than the right	x > y
<	Less that - True if left operand is less than the right	x < y
==	Equal to - True if both operands are equal	x == y
!=	Not equal to - True if operands are not equal	x != y
>=	Greater than or equal to - True if left operand is greater than or equal to the right	x >= y
<=	Less than or equal to - True if left operand is less than or equal to the right	x <= y

Comparison Operators (Relational Operators) Example

```
x = 10
y = 12
print('x > y  is',x>y)
print('x < y  is',x<y)
print('x == y is',x==y)
print('x != y is',x!=y)
print('x >= y is',x>=y)
print('x <= y is',x<=y)
```

```
Output
x > y  is False
x < y  is True
x == y is False
x != y is True
x >= y is False
x <= y is True
```

Bitwise operators

- Bitwise operators act on operands as if they were string of binary digits. It operates bit by bit, hence the name

Operator	Meaning
&	Bitwise AND
|	Bitwise OR
~	Bitwise NOT
^	Bitwise XOR
>>	Bitwise right shift
<<	Bitwise left shift

Assignment operators

Operator	Example	Equivalent to
=	x = 5	x = 5
+=	x += 5	x = x + 5
-=	x -= 5	x = x - 5
***=**	x *= 5	x = x * 5
/=	x /= 5	x = x / 5
%=	x %= 5	x = x % 5
//=	x //= 5	x = x // 5
****=**	x **= 5	x = x ** 5
&=	x &= 5	x = x & 5
|=	x |= 5	x = x | 5
^=	x ^= 5	x = x ^ 5
>>=	x >>= 5	x = x >> 5
<<=	x <<= 5	x = x << 5

Special Operators – Identity Operators

- is and is not are the identity operators in Python.
- They are used to check if two values (or variables) are located on the same part of the memory.
- Two variables that are equal does not imply that they are identical.

Example:

```
x1 = 5
y1 = 5
x2 = 'Hello'
y2 = 'Hello'
x3 = [1,2,3]
y3 = [1,2,3]
print(x1 is not y1)
print(x2 is y2)
print(x3 is y3)
```

Output

```
False
True
False
```

Special Operators – Membership Operators

in and not in are the membership operators in Python. They are used to test whether a value or variable is found in a sequence (string, list, tuple, set and dictionary). In a dictionary we can only test for presence of key, not the value.

Operator	Meaning	Example
in	True if value/variable is found in the sequence	5 in x
not in	True if value/variable is not found in the sequence	5 not in x

Special Operators – Membership Operators - Example

```
x = 'Hello world'
y = {1:'a',2:'b'}
print('H' in x)
print('hello' not in x)
print(1 in y)
print('a' in y)
```

```
Output
True
True
True
False
```

Questions