Lesson-3: Basic I/O, Datatypes, Literals and Identifiers

- Literals are nothing but constants that can be stored in memory. All literals can be grouped into various categories called data types.
- In Python, data types are decided dynamically by the interpreter. This is called dynamic typing.
- In Python, anything and everything is considered as an object.
- In Python, unlike other programming languages, variable names are kind of tag attached to the objects stored on the heap.

3 Types of Literals

- Numeric Literals
  - Integer Literal
  - Float Literal
  - Hexadecimal Literal
  - Octal Literal
  - Binary Literal
  - Complex Literal
- String Literals
- Boolean Literals
  - True
  - False

Built-in Data Types

- None
- Numeric
  - int
  - float
  - complex
- bool
- Sequences (Ordered)
  - str
  - bytes
  - bytearray
  - list
  - tuple
  - range
- Set
  - set
  - frozenset
- Map
  - dict

Program 1:

```
#This is my first python program

"""This is my first python
  program"""

'''This is my first
  python program'''

a = 5
b = +6

c = 1.0
d = +2.0

e = 5 + 2J
f = 2 + 3J

g = 0x1a
h = 0x23

i = 0b1010
j = 0b0001

k = 0o45
l = 0o46

m = True;

n = 'Hello World\n'
o = "Hello World\n"
p = '''Hello World\n'''
q =  """Hello World\n"""


sum_one   = a + b
sum_two   = c + d
sum_three = e + f
sum_four  = g + h
sum_five  = i + j
sum_six   = k + l

print("The sum of ", a, "and ", b , "is ", sum_one )
print("The sum of ", c, "and ", d , "is ", sum_two )
print("The sum of ", e, "and ", f , "is ", sum_three )
print("The sum of ", g, "and ", h , "is ", sum_four )
print("The sum of ", i, "and ", j , "is ", sum_five )
print("The sum of ", i, "and ", j , "is ", sum_six )
print(m)
print(n, o, p, q)
```

Output:

```
The sum of  5 and  6 is  11
The sum of  1.0 and  2.0 is  3.0
The sum of  (5+2j) and  (2+3j) is  (7+5j)
The sum of  26 and  35 is  61
The sum of  10 and  1 is  11
The sum of  37 and  38 is  75
True
Hello World
Hello World
Hello World
Hello World
```

User-defined Data Types

- These are those data types created by programmers

Program 2:

```
#Taking input through keyboard

a = input(“Enter first number\n”)
b = input(“Enter second number\n”)

c = input(“Enter third number\n”)
d = input(“Enter fourth number\n”)

sum_one = a + b
sum_two = c + d

print("The sum of ", a, "and ", b , "is ", sum_one )
print("The sum of ", c, "and ", d , "is ", sum_two )
```

Output:

```
Enter first number
23
Enter second number
45
Enter third number
56
Enter fourth number
67
The sum of  23 and  45 is  2345
The sum of  56 and  67 is  5667
```

Program 3:

```
#Taking input through keyboard

a = input("Enter first number\n")
b = input("Enter second number\n")

a = int(a)
b = int(b)

c = input("Enter third number\n")
d = input("Enter fourth number\n")

c = float(c)
d = float(d)

complex_number_one = complex(input("Enter first complex number"));
complex_number_two = complex(input("Enter second complex number"));

sum_one   = a + b
sum_two   = c + d
sum_three = complex_number_one + complex_number_two;

print("The sum of ", a, "and ", b , "is ", sum_one )
print("The sum of ", c, "and ", d , "is ", sum_two )
print("The sum of ", complex_number_one, "and ", complex_number_two , "is ", sum_three )
```

Output:

```
Enter first number
1
Enter second number
2
Enter third number
3
Enter fourth number
4
Enter first complex number
5+6j
Enter second complex number
4+5j
The sum of  1 and  2 is  3
The sum of  3.0 and  4.0 is  7.0
The sum of  (5+6j) and  (4+5j) is
(9+11j)
```

Guess The Output:

```
>>> 10        >>> -10        >>> .10        >>> 1,0        >>> 1,024

>>> 1, 0x20   >>> 1, 0o45    >>> 1, 0b1010

>>> print("hello")                    >>> print("hello", "hello")

>>> print("hello", "hello", sep='$')

>>> print("hello", "hello", sep='$', end='#')

>>> x = 15.6                >>> x = 15                >>> x = 0o17
>>> int(x)                  >>> float(15)              >>> int(x)

>>> x = 0B1110010           >>> x = 0x1c2             >>> x = "17"
>>> int(x)                  >>> int(x)                >>> int(x, 8)
```

NOTE: **int(x,8)** means convert the value of x which contains a value in octal format to integer.

```
>>> x = "1110010"           >>> x = "1c2"            >>> a = 10
>>> int(x,2)                >>> int(x,16)            >>> bin(a)

>>> b = 1010                >>> b = 10               >>> b = 10
>>> oct(b)                  >>> hex(10)              >>> type(b)
```

A BIG WARNING ABOUT FLOATING POINT REPRESENTATION: Range and Precision

Limits for floating Point Range = (10 to the power -308) to (10 to the power +308)

```
>>> 1.5e200 * 2.0e210  => inf (Arithmetic Overflow)
>>> 1.0e-300 / 1.0e100 => 0.0 (Arithmetic underflow)
```

Limits for floating Point Precision = 16 to 17 digits

```
>>> 1/3
0.3333333333333333 (This is an approximation to 16 digits)

>>> 3 * (1/3) [Will not result in 0.9999999999999999]
1.0 (Rounded Off)
```

>>> 1.9999999999999999
2.0 (Rounded Off)

>>> 1.9999999999999998
1.9999999999999998 (Not rounded off)

>>> 1.999999999999999 (1 fractional digit lesser than previous example)
1.9999999999999999 (Not rounded off)

More Examples:

>>> 1/10
0.1

>>> 6 * (1/10)
0.6000000000000001

>>> 10 * (1/10)
1.0

>>> 1/10 + 1/10 + 1/10
0.30000000000000004

MORAL OF THE STORY:

No matter how Python chooses to display calculated results, the value stored is limited in both the range of numbers that can be represented and the degree of precision. For most everyday applications, this slight loss in accuracy is of no practical concern. However, in scientific computing and other applications in which precise calculations are required, this is something that the programmer must be keenly aware of.

Program 4:

#formatting function

```
#formatting function

print(12/5 )
print(format(12/5, '0.2f'))
print(format(12/5, '0.3f'))
print(5/7)
print(format(5/7, '0.2f'))
print(format(11/12,'.3e'))
print(format(11/12,'.4e'))
print(format(11/12,'.2e'))
print(format(2**100,'.6e'))
```

Output:

```
2.4
2.40
2.400
0.7142857142857143
0.71
9.167e-01
9.1667e-01
9.17e-01
1.267651e+30
```

Guess The Output:

```
>>> print( 'Let's Go!' )
>>> print( 'Subhash' 'Python' 'Zebra' )
>>> print( 'Let's Go!' )
>>> print( "Lets Go!' )
>>> ord('A')                    #give the ascii value of
>>> ord('6')
>>> ord('65')
>>> ord(65)
>>> chr(65)
>>> chr('48')
>>> format('Hello', '<20')
>>> format('Hello', '>20')
>>> format('Hello', '^20')
>>> format('Hello', '.^20')
>>> format(65, '>20')
>>> print(format('-','-<20'),'Hello World', format('-','->20'))
>>> format( 1/3, ">20" )
```

More On Variables:

```
>>> n = 10
>>> n = n + 1
>>> n
11
>>> k = n
>>> k
11
>>> id(n)
123456
>>> id(k)
123456
>>> n = "Hello"
>>> n
Hello
>>> n = 12.5
>>> n
12.5
```

Guess The Output:

```
>>> n = 10
>>> id(n)
```

```
>>> print ("1""2""3")
```

```
>>> k = n
>>> id(k)
```

```
>>> print (1,2,3)
```

```
>>> n = n + 1
>>> id(n)
>>> id(k)
```

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

```
>>> print("Hello","Hello","Hello")
```

```
>>> print ("1"2"3")
```

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

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

Rules For Naming Variables:

- A variable name must start with a letter or the underscore character.
- A variable name cannot start with a number.
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Identify The Wrong Variable Names (or Identifiers):

a) iLoveIndia b) 1LoveIndia c)Love-India d)Love India e) _LoveIndia

f) and g) print h) Break

List Of Keywords In Python:

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

There are many predefined identifiers that can be used as regular identifiers, but should not be. This includes float, int, print, exit, and quit. To check if an identifier is part of built-in identifiers, then you can use the following command.

```
>>> 'exit' in dir(__builtins__)
```

```
>>> 'subhash' in dir(__builtins__)
```

```
>>> 'print' in dir(__builtins__)
```

Assignment Program

Restaurant Bill Calculation Program:

This program will calculate a restaurant bill for a couple with a gift coupon, with a restaurant tax of 18.0 %

Enter amount of the gift coupon:2000
Enter ordered items for person 1
Chicken Biriyani : 120
Chicken Ghee Roast: 160
Coke Drink : 25
Vanilla Ice Cream : 35

Enter ordered items for person 2
Vegetable Biriyani : 50
Gobi Manchurian : 30
Coke Drink : 25
Vanilla Ice Cream : 35
Bill without tax : 480.00
Total tax : 86.40
Total bill payable : 566.40
Amount balance in coupon : 1433.60