Python Input, Output And Import

- Python provides numerous built-in functions that are readily available to us at the Python prompt.

- Some of the functions like input() and print() are widely used for standard input and output operations respectively.

Python Output Using print() function

- We use the print() function to output data to the standard output device (screen).
- Syntax of the print() function is: ##### print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)
- Here, objects is the value(s) to be printed.
- The sep separator is used between the values. It defaults into a space character.
- After all values are printed, end is printed. It defaults into a new line.
- The file is the object where the values are printed and its default value is sys.stdout(screen).

In [17]:

```
print('Welcome to CCTIL Lab')
# Output: Welcome to CCTIL Lab

a = 5

print('The value of a is', a)
# Output: The value of a is 5
```

```
Welcome to CCTIL Lab
The value of a is 5
```

- In the second print() statement, we can notice that a space was added between the string and the value of variable a.This is by default, but we can change it.

In [18]:

```
print(1,2,3,4)
# Output: 1 2 3 4

print(1,2,3,4,sep='*')
# Output: 1*2*3*4

print(1,2,3,4,sep='#',end='&')
# Output: 1#2#3#4&
```

```
1 2 3 4
1*2*3*4
1#2#3#4&
```

Output formatting

Sometimes we would like to format our output to make it look attractive. This can be done by using the str.format() method. This method is visible to any string object.

- Syntax: **format(value[, format_spec])**

In [19]:

```
x = 5; y = 10
print('The value of x is {} and y is {}'.format(x,y))
```

```
The value of x is 5 and y is 10
```

- **Here the curly braces {} are used as placeholders. We can specify the order in which it is printed by using numbers (tuple index).**

In [20]:

```
print('I love {0} and {1}'.format('bread','butter'))
# Output: I love bread and butter

print('I love {1} and {0}'.format('bread','butter'))
# Output: I love butter and bread
```

```
I love bread and butter
I love butter and bread
```

We can even use keyword arguments to format the string.

In [21]:

```
print('Hello {name}, {greeting}'.format(greeting = 'Goodmorning', name = 'Students'))
```

```
Hello Students, Goodmorning
```

Python Input

- To allow flexibility we might want to take the input from the user.
- In Python, we have the <font color = "Red">input()</font>function to allow this.
- The syntax for input() is: **input([prompt])**

In [22]:

```
num = input('Enter a number: ')
```

```
Enter a number: 5
```

```
In [23]: print(num)
         print(type(num))

         5
         <class 'str'>
```

Here, we can see that the entered value 10 is a string, not a number. To convert this into a number we can use int() or float() functions.

```
In [24]: int('10')

Out[24]: 10
```

```
In [25]: float('10')

Out[25]: 10.0
```

This same operation can be performed using the eval() function. But it takes it further. It can evaluate even expressions, provided the input is a string

```
In [26]: int('2+3')

         ---------------------------------------------------------------------------
         ValueError                                Traceback (most recent call last)
         <ipython-input-26-506ddc2101a0> in <module>()
         ----> 1 int('2+3')

         ValueError: invalid literal for int() with base 10: '2+3'
```

```
In [ ]: eval('2+3')
```

Python Import

- When our program grows bigger, it is a good idea to break it into different modules.
- A module is a file containing Python definitions and statements. Python modules have a filename and end with the extension .py
- Definitions inside a module can be imported to another module or the interactive interpreter in Python. We use the import keyword to do this.
- For example, we can import the math module by typing in import math.

```
In [ ]: import math
        print(math.pi)
```

Now all the definitions inside math module are available in our scope. We can also import some specific attributes and functions only, using the from keyword. For example:

In [27]:

```
from math import pi
```

In [28]:

```
print(2*(math.pi))
```

```
6.283185307179586
```