1. Function: input()

1.1 To take input from the user use function **input**. Data entered by the user is returned as char/string data type.

```
# value assigned to variable "a" will always be in character/string data type
# even if the number is entered.
a = input('Enter any number:')
```

```
Enter any number:5
```

```
a
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-3f786850e387> in <module>()
----> 1 a

NameError: name 'a' is not defined
```

SEARCH STACK OVERFLOW

1.2 Type conversion functions like int(), float(), etc need to be used to get output in particular data type.

```
# value assigned to variable "b" is a float that is converted from character
# data type to float data type
b = float(input('Enter any number:'))
```

```
Enter any number:5
```

```
b
```

```
5.2
```

```
# value assigned to variable "c" is an integer that is converted from character
#  data type to integer data type
c = int(input('Enter any number:'))
```

```
Enter any number:5
```

Coercion:

It is an Implicit (automatic) conversion of an operand from one data type to another.

It is done if the operand is converted safely without any loss of information.

Eg. integer can be converted into float without any loss. EG. 4 ⇒ 4.0

But float value if converted to integer may end with information loss. Eg. 4.5 ⇒ 4

Type Conversion:

It is an explicit conversion of an operand from one data type to another.

It is done programmatically and may involve loss of information.

Built-in type conversion function available in python are: int() and float()

```
# float converted to integer
int(4.5)
```

```
4
```

```
# character converted to integer
```

```
int('4')
```

```
4
```

```
# character that represents real number cannot be converted to integer
int('4.5')
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-9-7f87cc7e2ff4> in <module>()
      1 # character that represents real number cannot be converted to integer
----> 2 int('4.5')

ValueError: invalid literal for int() with base 10: '4.5'
```

SEARCH STACK OVERFLOW

```
# character converted to float
float('4.5')
```

```
4.5
```

2. To Display Output

2.1 Function: print() : It is used to display the result to the user.

```
print('The result is')
```

```
The result is
```

```
print('The result is',a)
```

```
print('The result is',b)
```

```
The result is 5
The result is 5.2
```

```
# To display the value of variables a and b in between the text.
print("Hello %s to the %f " %(a,b))
```

```
Hello 5 to the 5.000000
```

```
# Another way to display the value of variables a and b in between the text.
#Sequence of the variables mentioned will be given a index value starting from 0
# This number can be mentioned inside the curly bracket and accordinly variable
# values will be displayed
print("Hello {0} to the {1} " )
```

```
Hello {0} to the {1}
```

```
# Sequence of the variables can be changed by mentioning the number inside the
# curly bracket
print("Hello {1} to the {0} " .format(a,b))
```

```
Hello 5.2 to the 5
```

```
# Text are displayed on two different lines by default
print("nishant")
print("jain")
```

```
nishant
jain
```

```
# To display text on same lines
```

```
print("nishant", end =" ")
print("jain")
```

```
nishant jain
```

```
# To display text on same lines separated by underscores
print("nishant", end ="_____")
print("jain")
```

```
nishant_____jain
```

2.2 Function format()

Syntax: format(value, format_specifier)

It can be used to format the output as per the requirement and to produce a numeric string version of the value containing a specific number of decimal places.

2.2.1 Example to display integer and float numbers:

```
# Displays value in the form of string upto 3 decimal places
format(11/4,'.3f')
```

```
'2.750'
```

```
# Displays value in the form of string upto 3 decimal places using space for
# total of 9 characters
# Since five character space is used by 2.750, remaining 4 spaces are represted
# by blank spaces on left side
print(format(11/4,'9.3f'))
```

```
    2.750
```

```
# Displays output in the form of exponential with accuracy upto 3 decimal places
format(2.0e100,0.3e)
```

```
  File "<ipython-input-13-705c2eaac345>", line 2
    format(2.0e100,0.3e)
                       ^
SyntaxError: invalid syntax
```

SEARCH STACK OVERFLOW

```
# Displays number with comma at proper place and with accuracy upto 3 decimal
# places
format(12345.23,',.3f')
```

```
'12,345.230'
```

```
# formal function can be used along with print function as given below:
print('The result is',format(b,'16.5f'))
```

2.2.2 Example to display string:

```
# It displays Hello in left justified format in a field width of 20 characters.
format('Hello','<20')
```

```
'Hello               '
```

```
# It displays Hello in Rught justified format in a field width of 20 characters.
format('Hello','>20')
```

```
'               Hello'
```

```
# It displays Hello in center aligned format in a field width of 20 characters.
format('Hello','^20')
```

```
'      Hello       '
```

```
# creates string of 10 blank characters.
format(' ','10')
```

```
'          '
```

```
# creates string with a as first character followed by 9 blank char.
format('a','10')
```

```
'a         '
```

```
# creates string with 9 a’s followed by 1 blank.
format(' ','a>10')
```

```
'aaaaaaaaa '
```

```
# creates string with 1 blank followed by 9 a’s.
format(' ','a<10')
```

```
' aaaaaaaaa'
```

```
# creates string with 1 blank at the center after 4 a’s.
format(' ','a^10')
```

```
'aaaa aaaaa'
```

```
# formal function can be used along with print function as given below:
print('The result is',format(a,'>10'))
```

```
The result is          5
```

Program statement line joining

Statements can be joined to form a proper python statement using the following two ways:

1. **Implicit Line Joining**: It uses Matching parenthesis, [] ,{} to join logical program line. Matching quotes must be on the same line.

Eg. print('Name', Employee_name, 'Age', employee_age)

2. **Explicit Line Joining**: It uses backslash (\) to join the lines.

Eg. print(a+b +
c*d)

[ ] ↳ *3 cells hidden*

Colab paid products - Cancel contracts here