Comments in Python

In [2]:

```
# this is the first comment
a = 10 # this is the second comment
#this is the third comment

str = "# this is not a comment, it is a quote!"
```

Using Python as a Calculator (Interactive Mode)

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory.

In [3]:

```
2+2
```

Out[3]:

```
4
```

In [4]:

```
48 - 33
```

Out[4]:

```
15
```

In [5]:

```
50 - 5*6
```

Out[5]:

```
20
```

In [6]:

```
(50 - 5*6) / 4
```

Out[6]:

```
5.0
```

In [7]:

```
8 / 5 # division always returns a floating point number
```

Out[7]:

```
1.6
```

In [8]:

```
# Classical division returns a float
9 / 4
```

Out[8]:

```
2.25
```

In [10]:

```
# Floor division discards the fractional part
9 // 4
```

Out[10]:

```
2
```

In [12]:

```
# the % operator returns the remainder of the division
9 % 4
```

Out[12]:

```
1
```

In [14]:

```
# Application of BODMAS Rule
(2 + 3) * 4
```

Out[14]:

```
20
```

In [15]:

```
# Square of a number
2 ** 5
```

Out[15]:

```
32
```

Variables

Variables are used to store information to be referenced and manipulated in a computer program.

In [16]:

```
# Defining a variable
width = 100
```

In [18]:

```
width # accessing the value of a variable
```

Out[18]:

```
100
```

In [19]:

```
# Computing using a variable
length = 40
area = width*length
print(area)
```

```
4000
```

In [20]:

```
# If a variable is uninitialized, trying to access it will throw an error.
```

In [21]:

```
# Type Conversion in Python(Integer -> Float)
4 * 3.75 - 1
```

Out[21]:

```
14.0
```

In [4]:

```
tax = 12.5/100
price = 250
tax_paid = price * tax
print(tax_paid)
```

```
31.25
```

In [7]:

```
a = 10
b = 20
a*b
```

Out[7]:

```
200
```

In [8]:

```
a+_ # stores value of last calculation
# In interactive mode, the last printed expression is assigned to the variable _
```

Out[8]:

```
210
```

In [9]:

```
b= 32.675432
b+_
```

Out[9]:

```
242.675432
```

In [10]:

```
round(_,2) # Rounding of a number upto 2 decimal places
```

Out[10]:

```
242.68
```

Strings

In [11]:

```
'I am Devjeet Roy' # single quoted text.
```

Out[11]:

```
'I am Devjeet Roy'
```

In [12]:

```
'I\'m Devjeet Roy' # Escape Sequence
```

Out[12]:

```
"I'm Devjeet Roy"
```

In [13]:

```
'"I am Fine!", he said.'
```

Out[13]:

```
'"I am Fine!", he said.'
```

In [15]:

```
s = 'First line \nSecondLine.'
s # without print(), \n is included in the output
```

Out[15]:

```
'First line \nSecondLine.'
```

In [16]:

```
print(s) # with print(), \n produces a new line
```

```
First line
SecondLine.
```

In [17]:

```
print('C:\some\name')  # here \n means newline!
```

```
C:\some
ame
```

In [19]:

```
print(r'C:\some\name')  # note the r before the quote, r is used to denote that it is a raw string
.
```

```
C:\some\name
```

In [22]:

```
print("""\
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
""")
# \ is used to remove the line which is taken during print for multi line activity.
#   End of lines are automatically included in the string, but it's possible to prevent.....
#   this by adding a \ at the end of the line.
```

```
Usage: thingy [OPTIONS]
     -h                        Display this usage message
     -H hostname               Hostname to connect to
```

In [23]:

```
# 3 times 'un', followed by 'ium'
3 * 'un' + 'ium'
```

```
# Print a part of the string 3 times, strictly follows the BODMAS Rule
```

Out[23]:

```
'unununium'
```

In [24]:

```
# Two or more string literals (i.e. the ones enclosed between quotes) next to each other are autom
atically concatenated.
'I am'' Devjeet'
# This feature is particularly useful when we want to break long strings.
```

Out[24]:

```
'I am Devjeet'
```

In [26]:

```
# But it has to be remembered that a variable and a string literal can't be concatenated by the ab
ove method.
prefix = 'Py'
prefix 'thon'
```

```
  File "<ipython-input-26-2cd011c62146>", line 3
    prefix 'thon'
           ^
SyntaxError: invalid syntax
```

In [27]:

```
prefix + 'thon'
```

Out[27]:

```
'Python'
```

In [28]:

```
word = "Devjeet"
word[4] # Indexing
```

Out[28]:

```
'e'
```

In [29]:

```
word[-1] # Negative Indexing.
```

Out[29]:

```
't'
```

In [31]:

```
# Slicing
word[3:7] # Exclusive of upper limit but inclusive of lower limit.
```

Out[31]:

```
'jeet'
```

In [32]:

```
word[:4]
```

Out[32]:

```
'Devj'
```

In [33]:

```
word[1:]
```

Out[33]:

```
'evjeet'
```

In [34]:

```
word[-4:]
```

Out[34]:

```
'jeet'
```

In [35]:

```
# It has to be remembered that Python Strings are immutable
word[0] ="T"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-35-ccad4296d73e> in <module>
      1 # It has to be remembered that Python Strings are immutable
----> 2 word[0] ="T"

TypeError: 'str' object does not support item assignment
```

In [36]:

```
# The alternative to above one is :
"T"+ word[1:]
```

Out[36]:

```
'Tevjeet'
```

In [37]:

```
s = "I am Devjeet Roy. I am a CSE Student."
len(s) # Returns length of the string.
```

Out[37]:

```
37
```

Introduction to List

Python knows a number of compound data types, used to group together other values. The most versatile is the list, which can be written as a list of comma-separated values (items) between square brackets. Lists might contain items of different types, but usually the items all have the same type.

In [38]:

```
letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

In [39]:

```
letters
```

Out[39]:

```
['a', 'b', 'c', 'd', 'e', 'f', 'g']
```

In [40]:

```
letters[2:5] = ['C', 'D', 'E'] # Replace Values
letters
```

Out[40]:

```
['a', 'b', 'C', 'D', 'E', 'f', 'g']
```

In [41]:

```
letters[2:5] = [] # Remove values
letters
```

Out[41]:

```
['a', 'b', 'f', 'g']
```

In [42]:

```
letters[:] = [] # Clears the whole list but keeps the list object
letters
```

Out[42]:

```
[]
```

In [43]:

```
MyName = ['D','e','v','j','e','e','t','R','o','y']
MyName
```

Out[43]:

```
['D', 'e', 'v', 'j', 'e', 'e', 't', 'R', 'o', 'y']
```

In [44]:

```
len(MyName) # Returns the length of the list
```

Out[44]:

```
10
```

In [45]:

```
a=['1','2','3']
b=['a','b','c']
finalList= [a,b]
finalList
```

Out[45]:

```
[['1', '2', '3'], ['a', 'b', 'c']]
```

In [46]:

```
finalList[1]
```

Out[46]:

```
['a', 'b', 'c']
```

In [47]:

```
finalList[1][2]
```

Out[47]:

```
'c'
```

In [ ]:

```

```