```
In [1]:  'Hello World'

Out[1]:  'Hello World'
```

```
In [15]: import sys
         import keyword
         import operator
         from datetime import datetime
         import os
```

```
In [ ]:  #  keywords are the reserved words in python and cant be used as an identifier
```

```
In [5]:  print(keyword.kwlist)

['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'clas
s', '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']
```

```
In [7]:  len(keyword.kwlist)

Out[7]:  35
```

```
In [28]: #''Identifiers
         # An identifier is a name given to entities like class, functions, variables, et
```

```
In [33]: 1var = 10 #identifier cant start with zero

  Cell In[33], line 1
    1var = 10 #identifier cant start with zero
    ^
SyntaxError: invalid decimal literal
```

```
In [35]: val12@ = 50 #identifier cant use special symbols

  Cell In[35], line 1
    val12@ = 50 #identifier cant use special symbols
           ^
SyntaxError: invalid syntax
```

```
In [37]: import = 125 #keywords are not allowed to use as an identifier

  Cell In[37], line 1
    import = 125 #keywords are not allowed to use as an identifier
           ^
SyntaxError: invalid syntax
```

```
In [51]: '''
         correct way of defining an identifier
         (Identifiers can be a combination of letters in lowercase (a to z) or uppercase
         '''
         val2 = 10
```

```
In [55]: val__ = 99
```

```
In [59]: val2
```

```
Out[59]:  10
```

```
In [61]:  val__
```

```
Out[61]:  99
```

```
In [63]:  '''
          Comments in Python
          comments can be used to explain the code for more readability
          '''
```

```
Out[63]:  '\nComments in Python\ncomments can be used to explain the code for more readab
          ility\n'
```

```
In [65]:  #single line comment
          val1 = 10
```

```
In [67]:  #multiple
          #line
          #comment
          val = 10
```

```
In [69]:  '''
          multiple
          line
          comment
          '''
          val1 = 10
```

```
In [71]:  '''
          multiple
          line
          comment
          '''
          val1 = 10
```

```
In [73]:  #Statements Instructions that a python interpreter can execute
```

```
In [77]:  p = 20 #create an integer object with value 20 and assign the variable p to p
          q = 20 #create new refernce q which will point to 20
          r = q #variable r will also point to same point where p and q are pointing
          p, type(p), hex(id(p)) #variable p is pointing to memory location
```

```
Out[77]:  (20, int, '0x7ff9ab4b3c18')
```

```
In [79]:  q, type(q), hex(id(q))
```

```
Out[79]:  (20, int, '0x7ff9ab4b3c18')
```

```
In [81]:  r, type(r), hex(id(r))
```

```
Out[81]:  (20, int, '0x7ff9ab4b3c18')
```

```
In [83]:  p = 20
          p = p + 10 #variable overwriting
          p
```

```
Out[83]:  30
```

```
In [1]:  #variable assignment
```

```
In [3]:  intvar = 20
         float = 2.2
         string = 'Hello'
         print(intvar)
         print(float)
         print(string)
```

```
20
2.2
Hello
```

```
In [1]:  intvar, floatvar, strvar = 10,2.57,'Python Language'
         print(intvar)
         print(floatvar)
         print(strvar)
```

```
10
2.57
Python Language
```

```
In [3]:  p1 = p2 = p3 = p4 = 44 #All variable pointing to the same value
         print(p1,p2,p3,p4)
```

```
44 44 44 44
```

```
In [20]: val1 = 10
         print(val1)
         print(type(val1))
         print(sys.getsizeof(val1))
         print(val1, " is Integer?", isinstance(val1, int))
```

```
10
<class 'int'>
28
10  is Integer? True
```

```
In [28]: val2 = 92.78 #float data type
         print(val2)
         print(type(val2))# type of object
         print(sys.getsizeof(val2))#size of float objects in bytes
         print(val2,'is float?', isinstance(val2,float)) #val2 is instance of value 2
```

```
92.78
<class 'float'>
24
92.78 is float? True
```

```
In [32]: val3 = 25 + 10j
         print(val3)
         print(type(val3))
         print(sys.getsizeof(val3))
         print(val3, " is complex?", isinstance(val3, complex))
```

```
(25+10j)
<class 'complex'>
32
(25+10j)  is complex? True
```

```
In [34]: sys.getsizeof(int())

Out[34]: 28

In [36]: sys.getsizeof(float())

Out[36]: 24

In [38]: sys.getsizeof(complex())

Out[38]: 32

In [40]: bool1 = True
         bool2 = False
         print(type(bool1))

         <class 'bool'>

In [42]: print(type(bool2))

         <class 'bool'>

In [44]: isinstance(bool1,bool)

Out[44]: True

In [46]: bool(0)

Out[46]: False

In [48]: bool(1)

Out[48]: True

In [50]: bool(None)

Out[50]: False

In [54]: bool(False)

Out[54]: False

 In [3]: #STRING CREATION

In [41]: str1 = 'Hello Python'
         print(str1)

         Hello Python

 In [5]: str = 'Hello world'
         print(str)

         Hello world

 In [7]: mystr = 'Hello world' #Define string using single quotes
         print(mystr)

         Hello world
```

```
In [9]:  mystr = "Hello world" #Deine string using double quotes
          print(mystr)

          Hello world

In [13]:  mystr = '''Hello
                      World''' #string defined using triple quotes
          print(mystr)

          Hello
                      World

In [21]:  mystr = ('Happy '
                   'Monday '
                   'Everyone  ')
          print(mystr)

          Happy Monday Everyone

In [27]:  mystr2 = 'Woohoo '
          mystr2 = mystr2*5
          mystr2

Out[27]:  'Woohoo Woohoo Woohoo Woohoo Woohoo '

In [29]:  len(mystr2) #Length of string

Out[29]:  35

In [34]:  #STRING INDEXING'''

In [43]:  str1

Out[43]:  'Hello Python'

In [45]:  str[0] #First character in string1

Out[45]:  'H'

In [49]:  str1[len(str1)-1]  #last character using length function in string1

Out[49]:  'n'

In [51]:  str1[-1]#Last character in string

Out[51]:  'n'

In [53]:  str1[6]#Fetch 7th element of the string

Out[53]:  'P'

In [55]:  str1[5]

Out[55]:  ' '

In [57]:  #String Slicing
In [61]: str[0:5] #string slicing fetch all characters from 0 to 5 index location

Out[61]: 'Hello'

In [65]: str1[6:12] #String slicing fetch all characters from 6 to 12 index location

Out[65]: 'Python'

In [67]: str1[-4:] #Retrieve last four character of the string

Out[67]: 'thon'

In [71]: str1[-6:] #Retrieve last six characters ofthe string

Out[71]: 'Python'

In [73]: str1[:6] #Retrieve first six characters of the string

Out[73]: 'Hello '

In [75]: # UPDATE AND DELETE STRING

In [77]: str1

Out[77]: 'Hello Python'

In [81]: #Strings are immutable which means elements of string can not be changed once th
         str1[0:5] = 'HOLA'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[81], line 2
      1 #Strings are immutable which means elements of string can not be changed
once they have assigned
----> 2 str1[0:5] = 'HOLA'

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

```
In [85]: del str1#delete a string
         print(str1)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[85], line 1
----> 1 del str1#delete a string
      2 print(str1)

NameError: name 'str1' is not defined
```

```
In [89]: #String Concatenation
         s1 = "Hello "
         s2 = "Rishabh"
         s3 = s1 + s2
         print(s3)

Hello Rishabh
```

In [93]:
```
txt = "  abcd efgh ijkl"
txt.lstrip()
```
Out[93]:
```
'abcd efgh ijkl'
```
In [95]:
```
txt = " abcd efgh ijkl"
txt.strip()
```
Out[95]:
```
'abcd efgh ijkl'
```
In [97]:
```
#Using escape escaltor
mystr = "My favourite TV series is "Game of Thrones""
```
```
  Cell In[97], line 2
    mystr = "My favourite TV series is "Game of Thrones""
                                        ^
SyntaxError: invalid syntax
```
In [101…
```
#using escape charactes to allow illeagal characters
mystr = "My favourite series is \"Game of Thrones\""
print(mystr)
```
```
My favourite series is "Game of Thrones"
```
In [103…
```
#List
#1) List is an ordered sequence of items.
#2) We can have different data types under a list. E.g we can have integer, floa
# a same list
```
In [105…
```
list1 = []
```
In [107…
```
print(type(list1))
```
```
<class 'list'>
```
In [109…
```
list2 = [10,30,60] #List of integers
```
In [111…
```
list3 = [10.77,30.66,60.89] #list of float numbers
```
In [113…
```
list4 = ['one', 'Two', 'Three'] #List of Strings
```
In [115…
```
list5 = ['Rishabh', 25, [50,100],[150,90]] #Nested List
```
In [121…
```
list6 = [100,'Rishabh', 17.65] #list of mixed datatypes
```
In [129…
```
list7 = ['Rishabh',25,[50,100],[150,90],{'John','David'}]
```
In [135…
```
len(list6) #length of list
```
Out[135…
```
3
```
In [137…
```
#LIST INDEXING
```
In [143…
```
list2[0] #retrieve first element of the list
```
Out[143…
```
10
```

```
In [145…  list4[0] #retrieve first element of the list

Out[145…  'one'

In [147…  list4[0][0] #Nested indexing - Access the first character of the first list elem

Out[147…  'o'

In [149…  list4[-1] #last element of the list

Out[149…  'Three'

In [153…  list5[-1] #last element of the list

Out[153…  [150, 90]

In [155…  #List Slicing

In [157…  mylist = ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight']

In [159…  mylist[0:3] #Return all item from 0 to 3rd index location excluding the item

Out[159…  ['One', 'Two', 'Three']

In [163…  mylist[2:5] #Return all items from 2nd to 5th index

Out[163…  ['Three', 'Four', 'Five']

In [165…  mylist[:3]

Out[165…  ['One', 'Two', 'Three']

In [167…  mylist[:2]

Out[167…  ['One', 'Two']

In [171…  mylist[-3:]

Out[171…  ['Six', 'Seven', 'Eight']

In [173…  mylist[-2:]

Out[173…  ['Seven', 'Eight']

In [175…  mylist[-1]

Out[175…  'Eight'

In [177…  mylist[:]

Out[177…  ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight']

In [179…  #Add remove change items

In [181…  mylist
```

```
Out[181…  ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight']
```

```
In [187…  mylist.append('nine') #Append will add item at the end of the list
```

```
In [185…  mylist
```

```
Out[185…  ['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'nine']
```

```
In [191…  mylist.insert(9,'Ten') #Add item at the index levelof 9
          mylist
```

```
Out[191…  ['One',
           'Two',
           'Three',
           'Four',
           'Five',
           'Six',
           'Seven',
           'Eight',
           'nine',
           'Ten',
           'Ten',
           'nine']
```

```
In [205…  mylist.insert(1, 'ONE') #Add item at index location 1
          mylist
```

```
Out[205…  ['One',
           'ONE',
           'Two',
           'Three',
           'Four',
           'Five',
           'Six',
           'Seven',
           'Eight',
           'nine',
           'Ten',
           'Ten',
           'nine']
```

```
In [207…  mylist.remove('ONE')#remove item 'ONE'
          mylist
```

```
Out[207…  ['One',
           'Two',
           'Three',
           'Four',
           'Five',
           'Six',
           'Seven',
           'Eight',
           'nine',
           'Ten',
           'Ten',
           'nine']
```

```
In [210…  mylist.pop() #Remove last item of the list
          mylist
```

Out[210…
```
['One',
 'Two',
 'Three',
 'Four',
 'Five',
 'Six',
 'Seven',
 'Eight',
 'nine',
 'Ten',
 'Ten']
```

In [212…
```python
mylist.pop(8) #Remove item located at 8th index of list
mylist
```

Out[212…
```
['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight', 'Ten', 'Ten']
```

In [214…
```python
del mylist[7] #Remove item at index location 7 of list
mylist
```

Out[214…
```
['One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Ten', 'Ten']
```

In [216…
```python
#change value of the string
mylist[0] = 1
mylist[1] = 2
mylist[2] = 3
mylist
```

Out[216…
```
[1, 2, 3, 'Four', 'Five', 'Six', 'Seven', 'Ten', 'Ten']
```

In [218…
```python
mylist.clear() #Empty list/Delete all items in the list
mylist
```

Out[218…
```
[]
```

In [220…
```python
del mylist #Delete the whole list
mylist
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[220], line 2
      1 del mylist #Delete the whole list
----> 2 mylist

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

In [222…
```python
#Copy List
mylist = ['one','two','three','four','five','six','seven','eight','nine']
mylist1 = mylist #create a new reference "mylist1"
```

In [226…
```python
id(mylist), id(mylist1) #The adress of bot mylist willbe same
```

Out[226…
```
(1633430680000, 1633430680000)
```

In [228…
```python
mylist2 = mylist.copy() #create a copy of list
```

In [230…
```python
id(mylist2) #the adress will be different
```

```
Out[230…  1633429779136
```

```
In [232…  mylist[0] = 1
          mylist
```

```
Out[232…  [1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

```
In [236…  mylist1 #mylist1 will be also impacted as it is pointing to same lists
```

```
Out[236…  [1, 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

```
In [240…  mylist2 #copy of list wont be imapcted due to changes in original list
```

```
Out[240…  ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight', 'nine']
```

```
In [242…  #Join lists
          list1 = ['one', 'two', 'three', 'four']
          list2 = ['five', 'six', 'seven', 'eight']
          list3 = list1 + list2 #join two lists by + operator
          list3
```

```
Out[242…  ['one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight']
```

```
In [244…  list1.append(list2) #append list 2 with list2
          list1
```

```
Out[244…  ['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]
```

```
In [246…  #list membership
```

```
In [248…  list1
```

```
Out[248…  ['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]
```

```
In [250…  'one' in list1 #check if one exist in list
```

```
Out[250…  True
```

```
In [254…  'ten' in list1 #check if ten exist in list
```

```
Out[254…  False
```

```
In [258…  if 'three' in list1: #check if three is present in list
              print('Three is present in list ')
          else:
              print('Three is not present in list')
```

```
Three is present in list
```

```
In [262…  if 'eleven' in list1: #check if eleven is present in list
              print('Eleven is present in list')
          else:
              print('Eleven is not present in list')
```

```
Eleven is not present in list
```

```
In [264…  #Reverse and sort list
```

In [266…
```
list1
```
Out[266…
```
['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]
```
In [272…
```
list1.reverse() #reverse the list
```
In [270…
```
list1
```
Out[270…
```
[['five', 'six', 'seven', 'eight'], 'four', 'three', 'two', 'one']
```
In [280…
```
list1 = list1[::-1] #Reverse the List
list1
```
Out[280…
```
['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]
```
In [286…
```
mylist3 = [9,5,2,99,12,88,34]
mylist3.sort() #sort list in ascending order
```
In [284…
```
mylist3
```
Out[284…
```
[2, 5, 9, 12, 34, 88, 99]
```
In [290…
```
mylist3 = [9,5,2,99,88,34]
mylist3.sort(reverse=True) #sort in descending order
mylist3
```
Out[290…
```
[99, 88, 34, 9, 5, 2]
```
In [294…
```
mylist4 = [88,65,33,21,11,98]
sorted(mylist4) #retirns new sorted list and does not change the original
```
Out[294…
```
[11, 21, 33, 65, 88, 98]
```
In [296…
```
mylist4
```
Out[296…
```
[88, 65, 33, 21, 11, 98]
```
In [298…
```
#Loop through a list
```
In [300…
```
list1
```
Out[300…
```
['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]
```
In [302…
```
for i in list1:
    print(i)
```
```
one
two
three
four
['five', 'six', 'seven', 'eight']
```
In [306…
```
for i in enumerate(list1):
    print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, ['five', 'six', 'seven', 'eight'])
```

In [308…
```
#Count
```

In [310…
```
list10 = ['one','two','three','four','one','one','two','three']
```

In [312…
```
list10.count('one') #number of times item one is occured in the list
```

Out[312…
```
3
```

In [314…
```
list10.count('two')
```

Out[314…
```
2
```

In [316…
```
list10.count('four')
```

Out[316…
```
1
```

In [320…
```
# All/ANY The all() method returns:
# True - If all elements in the lists are true
#False - If any element in the list is false
# The any() function returns True if any elements in the  list is True, If not a
```

In [322…
```
l1 = [1,2,3,4,0]
```

In [326…
```
all(l1) #will return false as one value is false(vale0)
```

Out[326…
```
False
```

In [330…
```
any(l1) #will return true as we have item in the list with true values
```

Out[330…
```
True
```

In [336…
```
l2 = [1,2,3,4,True,False]
all(l2) # Returns false as one value is false
```

Out[336…
```
False
```

In [340…
```
any(l2)  # Will Return True as we have items in the list with True value
```

Out[340…
```
True
```

In [342…
```
l3 = [1,2,3,True]
```

In [346…
```
all(l3) # Will return True as all items in the list are True
```

Out[346…
```
True
```

In [366…
```
x = 10
y = 5
z = x + y
```

```
In [370…  print(z)

          15

In [372…  type(z)

Out[372…   int

In [9]:   x = 5
          y = 2
          print(x//y)
          type(y)

          2

Out[9]:    int

In [386…  str = 'hello'
          str1 = 'world'
          str2 = str + str1
          print(str2)

          helloworld

In [394…  x = "Hello"
          print(x[1:4])

          ell

In [396…  list1

Out[396…   ['one', 'two', 'three', 'four', ['five', 'six', 'seven', 'eight']]

In [411…  x = [4,7,1,12,3]
          largest_number = max(x)
          print(largest_number)

          12

In [12]:  x = [1,2,3,4]
          y = filter(lambda a:a % 2 ==0,x)
          print(list(y))

          [2, 4]

In [14]:  x = 20
          print(x)

          20

In [16]:  float(x)

Out[16]:   20.0

In [25]:  """ This is a multiline string in python """

Out[25]:   ' This is a multiline string in python '

In [48]:  dic = {1: 'A', 2: 'E', 3: 'I'}
          dic[4] = 'O'
          print(dic)
```

```
{1: 'A', 2: 'E', 3: 'I', 4: 'O'}
```

In [59]:
```
list1 = ['a', 'b', 'g', 1, 5]
print(list1)
```

```
['a', 'b', 'g', 1, 5]
```

In [63]:
```
print(list1.pop())
```

```
5
```

In [66]:
```
list1
```

Out[66]:
```
['a', 'b', 'g', 1]
```

In [68]:
```
list1.append(5)
```

In [70]:
```
list1
```

Out[70]:
```
['a', 'b', 'g', 1, 5]
```

In [72]:
```
var = 2
print(2 ==2.0)
```

```
True
```

In [74]:
```
num = 4 + 0j
print(type(num))
```

```
<class 'complex'>
```

In [76]:
```
print(int(3.9))
```

```
3
```

In [78]:
```
a = 'Pyhthon' + ".py"
print(a)
```

```
Pyhthon.py
```

Tuple Creation

In [83]:
```
tup1 = () #empty tuple
```

In [85]:
```
tup2 = (10,30,60) #tuple of integers numers
```

In [87]:
```
tup3 = (10.77,0.66,60.89) #tuple of float numbers
```

In [89]:
```
tup4 = ('one','two','three') #tuple of strings
```

In [91]:
```
tup5 = ('Asif',25,(50,100),(150,90)) #nested tuples
```

In [93]:
```
tup6 = (100,'Asif',17.765) #tuple of mixed datatypes
```

In [95]:
```
tup7 = ('Asif',25,[50,100],[150,90],{'John','David'},(99,22,33))
```

In [97]:
```
len(tup7)
```

Out[97]:
```
6
```

Tuple Indexing

```
In [102…  tup2[0]#retrieve the first element of the tuple
Out[102…  10
```

```
In [108…  tup4[0]#retrieve the first element of the tuple
Out[108…  'one'
```

```
In [112…  tup4[0][0] #nested indexing - access the first character of the first tuple
Out[112…  'o'
```

```
In [120…  tup4[-1] #last item of the tuple
Out[120…  'three'
```

```
In [122…  tup5[-1] #last item of the tuple
Out[122…  (150, 90)
```

Tuple Slicing

```
In [126…  mytuple = ('one','two','three','four','five','six','seven','eight')
```

```
In [128…  mytuple[0:3]#return all item from index 0 to 3
Out[128…  ('one', 'two', 'three')
```

```
In [130…  mytuple[2:5] #return all item from index 2 to 5
Out[130…  ('three', 'four', 'five')
```

```
In [132…  mytuple[:3] #return the first 3 items
Out[132…  ('one', 'two', 'three')
```

```
In [134…  mytuple[:2] #return the first 2 element
Out[134…  ('one', 'two')
```

```
In [138…  mytuple [-3:] #return last 3 item
Out[138…  ('six', 'seven', 'eight')
```

```
In [140…  mytuple[-2:]
Out[140…  ('seven', 'eight')
```

```
In [146…  mytuple[-1]
Out[146…  'eight'
```

```
In [148…  mytuple[:]

Out[148…  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [150…  del mytuple[0]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[150], line 1
----> 1 del mytuple[0]

TypeError: 'tuple' object doesn't support item deletion
```

```
In [152…  mytuple[0] = 1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[152], line 1
----> 1 mytuple[0] = 1

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

```
In [154…  del mytuple

In [156…  mytuple
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[156], line 1
----> 1 mytuple

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

Loop Through a tuple

```
In [162…  mytuple = ('one','two','three','four','five','six','seven','eight')

In [164…  mytuple

Out[164…  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [166…  for i in mytuple:
              print(i)
```

```
one
two
three
four
five
six
seven
eight
```

```
In [168…  for i in enumerate(mytuple):
              print(i)
```

```
(0, 'one')
(1, 'two')
(2, 'three')
(3, 'four')
(4, 'five')
(5, 'six')
(6, 'seven')
(7, 'eight')
```

```
In [170…  'one' in mytuple

Out[170…  True

In [172…  'ten' in mytuple

Out[172…  False

In [174…  if 'three' in mytuple:
              print('Three is not present in the tuple')
          else:
              print('Three is not present in the tuple')

          Three is not present in the tuple

In [ ]:
```

INDEX POSITION

```
In [178…  mytuple

Out[178…  ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [182…  mytuple.index('one') #index of first element equal to  'one'

Out[182…  0

In [184…  mytuple.index('five') #index of first element equal to five

Out[184…  4

In [189…  mytuple1 = ('one','two','three','four','one','one','two','three')

In [191…  mytuple1

Out[191…  ('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')

In [193…  mytuple1.index('one')#index of first element equal to 'one'

Out[193…  0

In [195…  #Sorting

In [3]:   mytuple2 = (43,67,99,12,6,90,67)

In [199…  sorted(mytuple2)
```

```
Out[199…  [6, 12, 43, 67, 67, 90, 99]
```

```
In [5]:  sorted(mytuple2, reverse=True)
```

```
Out[5]:  [99, 90, 67, 67, 43, 12, 6]
```

```
In [ ]:
```

SET

```
In [5]:  s = {}
```

```
In [7]:  type(s)
```

```
Out[7]:  dict
```

```
In [9]:  s = set()
         type(s)
```

```
Out[9]:  set
```

```
In [11]:  s.add(10)
```

```
In [13]:  s
```

```
Out[13]:  {10}
```

```
In [15]:  s.add(10,20)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 s.add(10,20)

TypeError: set.add() takes exactly one argument (2 given)
```

```
In [17]:  s.add(20)
```

```
In [19]:  s
```

```
Out[19]:  {10, 20}
```

```
In [21]:  s.add(30)
          s.add(40)
          s.add(50)
```

```
In [23]:  s
```

```
Out[23]:  {10, 20, 30, 40, 50}
```

```
In [25]:  len(s)
```

```
Out[25]:  5
```

```
In [27]: s[:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[27], line 1
----> 1 s[:]

TypeError: 'set' object is not subscriptable
```

```
In [36]: s
```

```
Out[36]: {10, 20, 30, 40, 50}
```

```
In [38]: s[3:]
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[38], line 1
----> 1 s[3:]

TypeError: 'set' object is not subscriptable
```

```
In [40]: s
```

```
Out[40]: {10, 20, 30, 40, 50}
```

```
In [42]: s.add(10)
```

```
In [44]: s
```

```
Out[44]: {10, 20, 30, 40, 50}
```

```
In [46]: s1 = set()
         s1
```

```
Out[46]: set()
```

```
In [48]: s1.add(2)
         s1.add(5.6)
         s1.add('nit')
         s1.add(1+2j)
         s1.add(True)
```

```
In [50]: s1
```

```
Out[50]: {(1+2j), 2, 5.6, True, 'nit'}
```

```
In [52]: s
```

```
Out[52]: {10, 20, 30, 40, 50}
```

```
In [54]: s2 = s.copy()
```

```
In [56]: s2
```

```
Out[56]:  {10, 20, 30, 40, 50}

In [60]:  s3 = set()
          s3

Out[60]:  set()

In [69]:  s3.add(100)
          s3.add(2)
          s3.add(15)
          s3.add(95)


In [72]:  s3

Out[72]:  {2, 15, 95, 100}

In [74]:  s[1:]

        ---------------------------------------------------------------------------
        TypeError                                 Traceback (most recent call last)
        Cell In[74], line 1
        ----> 1 s[1:]

        TypeError: 'set' object is not subscriptable

In [76]:  s3

Out[76]:  {2, 15, 95, 100}

In [78]:  len(s3)

Out[78]:  4

In [80]:  s3.clear()

In [82]:  s3

Out[82]:  set()

In [84]:  s2

Out[84]:  {10, 20, 30, 40, 50, 95}

In [86]:  s2.pop()

Out[86]:  50

In [88]:  s2

Out[88]:  {10, 20, 30, 40, 95}

In [90]:  s

Out[90]:  {10, 20, 30, 40, 50}
```

```
In [92]:  s1

Out[92]:  {(1+2j), 2, 5.6, True, 'nit'}

In [94]:  s1.pop()

Out[94]:  True

In [96]:  s1.remove((1+2j))

In [98]:  s1

Out[98]:  {2, 5.6, 'nit'}

In [100…  s2

Out[100…  {10, 20, 30, 40, 95}

In [106…  s2.add(100)

In [108…  s2

Out[108…  {10, 20, 30, 40, 95, 100}

In [110…  s2.remove(100)
          s2

Out[110…  {10, 20, 30, 40, 95}

In [112…  100 in  s2

Out[112…  False

In [114…  10 in s2

Out[114…  True

In [116…  s2

Out[116…  {10, 20, 30, 40, 95}

In [118…  s2.discard(100)

In [120…  s2

Out[120…  {10, 20, 30, 40, 95}

In [122…  s2.discard(20)
          s2

Out[122…  {10, 30, 40, 95}

In [128…  s2.remove(40)
In [131…  s2
Out[131…  {10, 30, 95}
```

for i in s2: print(i)

```
In [137…  for i in enumerate(s2):
            print(i)
(0, 10)
(1, 30)
(2, 95)
```

SET OPERATION

```
In [148…  A = {1,2,3,4,5}
           B = {4,5,6,7,8}
           C = {8,9,10}

In [144…  A.union(B)
Out[144…  {1, 2, 3, 4, 5, 6, 7, 8}

In [150…  B.union(C)
Out[150…  {4, 5, 6, 7, 8, 9, 10}

In [152…  A | B
Out[152…  {1, 2, 3, 4, 5, 6, 7, 8}

In [154…  A | B | C
Out[154…  {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

In [156…  print(A,B,C)
{1, 2, 3, 4, 5} {4, 5, 6, 7, 8} {8, 9, 10}

In [158…  print(A)
           print(B)
           print(C)
{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}

In [160…  A.intersection(C)
Out[160…  set()

In [162…  B & C
Out[162…  {8}
```

```
In [164…  print(A)
          print(B)
          print(C)

{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}

In [166…  A.difference(B)

Out[166…  {1, 2, 3}

In [168…  B.difference(A)

Out[168…  {6, 7, 8}

In [170…  A.difference(C)

Out[170…  {1, 2, 3, 4, 5}

In [172…  C.difference(A)

Out[172…  {8, 9, 10}

In [174…  A - B

Out[174…  {1, 2, 3}

In [176…  B - C

Out[176…  {4, 5, 6, 7}

In [178…  print(A)
          print(B)
          print(C)

{1, 2, 3, 4, 5}
{4, 5, 6, 7, 8}
{8, 9, 10}

In [180…  A.symmetric_difference(B)

Out[180…  {1, 2, 3, 6, 7, 8}

In [184…  B.symmetric_difference(C)

Out[184…  {4, 5, 6, 7, 9, 10}

In [186…  A & B

Out[186…  {4, 5}

In [188…  A

Out[188…  {1, 2, 3, 4, 5}
```

```
In [3]: 10
        print('hello')
```

```
hello
```

```
In [7]: 10 = emp_id
        print(emp_id)
```

```
  Cell In[7], line 1
    10 = emp_id
    ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

```
In [11]: if = 100
         print(if)
```

```
  Cell In[11], line 1
    if = 100
       ^
SyntaxError: invalid syntax
```

```
In [15]: a
         print(a)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[15], line 1
----> 1 a
      2 print(a)

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

```
In [17]: x = {10, 20, 30, 40}
         print(type(x))
```

```
<class 'set'>
```

```
In [19]:  y = {1: "Deva", 2: "Kumari", 3: "Prasad", 4: "Mani"}
         print(type(y))
```

```
<class 'dict'>
```

```
In [21]: a = 45
         print(a)
         a += 5
         print(a)
```

```
45
50
```

```
In [23]:  x = "Hello World"
         print(x[0:7])
```

```
Hello W
```

```
In [25]: s = "Python programming language"

         n = s.split()

         print(n)
```

```
['Python', 'programming', 'language']
```

```
In [27]: str1 = 'hello'
         print(str1[-1::])
```

```
o
```

```
In [29]: str1="hello"

         print(str1[::-1])
```

```
olleh
```

```
In [33]:  x = "python"
          x[3] = 's'

          print(x)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[33], line 2
      1 x = "python"
----> 2 x[3] = 's'
      4 print(x)

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

```
In [35]: print("ABCDEF".upper())
```

```
ABCDEF
```

```
In [37]:  a = [10, 20, 30]
         print(a*2)
```

```
[10, 20, 30, 10, 20, 30]
```

```
In [39]: a = 2a**2
```

```
  Cell In[39], line 1
    a = 2a**2
        ^
SyntaxError: invalid decimal literal
```

```
In [41]: a = [10, 20, 30]
         print(a**2)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[41], line 2
      1 a = [10, 20, 30]
----> 2 print(a**2)

TypeError: unsupported operand type(s) for ** or pow(): 'list' and 'int'
```

```
In [45]: values = [10, 20, 30, 40, 50, 60, 70, 80, 90]
         result = [value for value in values if value <= 50]

         print(result)
```

```
[10, 20, 30, 40, 50]
```

```
In [47]: name = ("python", )
         print(type(name))
```

```
<class 'tuple'>
```

In [49]:
```python
 t = (10, 20, 30, 40, 50, 60)
print(t[2:100])
```

```
(30, 40, 50, 60)
```

In [51]:
```python
t = (10, 20, 30)
print(t.index(30))
```

```
2
```

In [53]:
```python
 r = range(0, 10)
 x = set(r)
 print(x)
```

```
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9}
```

In [55]:
```python
s = {}
print(type(s))
```

```
<class 'dict'>
```

In [57]:
```python
 n = {10, 20, 30, 40, 50, 10, 10, 10}
print(len(n))
```

```
5
```

In [59]:
```python
s = {x*x  for x in range(5)}
print(s)
```

```
{0, 1, 4, 9, 16}
```

In [61]:
```python
d = {100: "Ramesh", 200: "Suresh", 300: "Mohan"}
print(d.keys())
```

```
dict_keys([100, 200, 300])
```

In [63]:
```python
 squares = {a:  a*a for a in range(1,6)}
print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [65]:
```python
2 * 2 * 2 * 2 * 2
```

Out[65]:
```
32
```

In [67]:
```python
2 ** 5
```

Out[67]:
```
32
```

Print Function

In [2]:
```python
# print is use for answers
```

In [4]:
```python
a = 10
b = 20
a
b
```

```
Out[4]:  20
```

```
In [6]:  a = 10
         b = 20
         print(a)
         print(b)
```

```
10
20
```

```
In [8]:  print(10)
         print(10,20)
         print('python')
         print(10,20,'python')
```

```
10
10 20
python
10 20 python
```

```
In [10]:  num1 = 20
          num2 = 30
          add = num1 + num2
          print(add)
```

```
50
```

Print Result with string

```
In [19]:  num1 = 20
          num2 = 30
          add = num1 + num2
          print('The addition of',num1,'and',num2,'is='  , add)
```

```
The addition of 20 and 30 is= 50
```

```
In [21]:  name = 'Python'
          age = 20
          city = 'Hyderabad'
```

```
In [23]:  print('My name is',name,'and i am',age,'years old form',city)
```

```
My name is Python and i am 20 years old form Hyderabad
```

```
In [25]:  ## Print format method
```

```
In [27]:  num1 = 20
          num2 = 30
          add = num1 + num2
          print('The addition of {} and {} is = {}'.format(num1,num2,add))
```

```
The addition of 20 and 30 is = 50
```

```
In [29]:  name = 'Python'
          age = 20
          city = 'hyd'
          #hellow my name is python and i am 10 year old from hyderabad
```

```
In [35]: print('hello my name is {}, and i am {} years old from {}'
         .format(name,age,city))
```

```
hello my name is Python, and i am 20 years old from hyd
```

```
In [39]: num1 = 100
         num2 = 25
         num3 = 333
         avg = (num1+num2+num3)/3 #or we can use avg=round(num1+num2+num3)/3,2)
         avg1 = round((num1+num2+num3)/3,2)
         print('The average of {},{},and {} is = {}'.format(num1,num2,num3,avg,avg1))#her
```

```
The average of 100,25,and 333 is = 152.66666666666666
```

```
In [41]: round(avg,2) #round of till 2 digits after decimal

Out[41]: 152.67
```

```
In [43]: # More short format meythod(f string method)
         # variable should be in curly braces
         # and write everything inside quots ''
         # at starting simpaly add f
```

```
In [45]: num1 = 20
         num2 = 30
         add = num1+num2
         print(f'The addition of {num1} and {num2} is = {add}')
```

```
The addition of 20 and 30 is = 50
```

```
In [47]: name = 'Python'
         age = 20
         city = 'Hyderabad'
         #hello my name is python and  i am 10 years old from hyderabad
```

```
In [49]: print(f'hello my name is {name}, and i am {age} year old, from {city}.')
```

```
hello my name is Python, and i am 20 year old, from Hyderabad.
```

```
In [51]: num1 = 10
         num2 = 20
         add = num1 + num2
         print('The addition of',num1,'and',num2,'is=',add)
         print('The addition of {} and {} is = {}'.format(num1,num2,add))
         print(f'The addition of {num1} and {num2} is={add}')
```

```
The addition of 10 and 20 is= 30
The addition of 10 and 20 is = 30
The addition of 10 and 20 is=30
```

End Statement

```
In [54]: print('hello') #1st statement
         print('good morning') #2nd statement
```

```
hello
good morning
```

Separator

```
In [58]: #here one print statement only we use
         #insisde one print statement we have multipal values
         #we want to seperate these multipal values with anything
```

```
In [60]: print('hello','Hii','How are you',sep='--->')
```

```
hello--->Hii--->How are you
```

```
In [63]: print('hello','hii','how are you',sep='&')
```

```
hello&hii&how are you
```

```
In [65]: print('hello','hii','how are you',sep='@')
```

```
hello@hii@how are you
```

```
In [73]: print('hello','Hii','how are you',sep=' ')
```

```
hello Hii how are you
```

```
In [75]: print(3,'.') # . is far from 3 so here we will use sep method
```

```
3 .
```

```
In [77]: print(3,'.',sep='') #see now space settled(also use to remove spcae b/w words)
```

```
3.
```

```
In [79]: print(1,2,end = ' ')
         print(3,'.',sep='')
         #will print 1 2 3.
```

```
1 2 3.
```

```
In [ ]:
```