In [3]:
```
print("hello world")
```
```
hello world
```

Variables

In [93]:
```
x = 100
#x is a variable and the number is a value
```

In [94]:
```
x
```
Out[94]:
```
100
```

Rules for naming variables

In [82]:
```
x=100
```

In [83]:
```
_x=100
```

In [84]:
```
1x=100
```
```
  Input In [84]
    1x=100
     ^
SyntaxError: invalid syntax
```

In [85]:
```
@x=100
```
```
  Input In [85]
    @x=100
      ^
SyntaxError: invalid syntax
```

In [86]:
```
x10=100
```

In [87]:
```
_x10=100
```

In [88]:
```
x-10=100
```

```
  Input In [88]
    x-10=100
    ^
SyntaxError: cannot assign to operator
```

In [92]:
```
x10@=100
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [92], in <cell line: 1>()
----> 1 x10@=100

TypeError: unsupported operand type(s) for @=: 'int' and 'int'
```

In [97]:
```
#case sensitive
a=100
A=300
```

In [98]:
```
a
```

Out[98]:
```
100
```

In [99]:
```
A
```

Out[99]:
```
300
```

In [100]:
```
#cannot use reserved words as variable name
break=10
```

```
  Input In [100]
    break=10
         ^
SyntaxError: invalid syntax
```

Data types

integer

In [13]:
```
x = 100
```

In [14]:
```
x
```

Out[14]:
```
100
```

In [15]:
```
type(x)
```
Out[15]:
```
int
```

floats

In [16]:
```
y = 10.2
```
In [18]:
```
y
```
Out[18]:
```
10.2
```
In [19]:
```
type(y)
```
Out[19]:
```
float
```

strings

In [23]:
```
z="hi,vaishnavi here"
```
In [24]:
```
z
```
Out[24]:
```
'hi,vaishnavi here'
```
In [25]:
```
type(z)
```
Out[25]:
```
str
```

Data types sepcific to python

list

In [27]:
```
l=[12,23,34,56]
```
In [28]:
```
type(l)
```
Out[28]:
```
list
```
In [30]:
```
l
```
Out[30]:
```
[12, 23, 34, 56]
```
In [38]:
```
#if you want to extract any one character you can call it by index starting
l[0]
```
Out[38]:
```
12
```

In [39]:
```
l[3]
```
Out[39]:
```
56
```

In [44]:
```
#if you want to change any character in list you can change it using its ind
l[2]=98
```

In [47]:
```
#previously l[2] was 34 now as we changed its values it have updated to late
l
```
Out[47]:
```
[12, 23, 98, 56]
```

tuples

In [48]:
```
t=(14,34,56,76)
```

In [49]:
```
type(t)
```
Out[49]:
```
tuple
```

In [50]:
```
t[2]
```
Out[50]:
```
56
```

In [52]:
```
#difference between list and tuple
#1.list have square brackets[] whereas,tuple have normal brackets()
#2.in list we can change its value(mutable) and in tuple we cannot change it
```

In [53]:
```
l
```
Out[53]:
```
[12, 23, 98, 56]
```

In [54]:
```
l[2]=9000
```

In [55]:
```
l
```
Out[55]:
```
[12, 23, 9000, 56]
```

In [56]:
```
t
```
Out[56]:
```
(14, 34, 56, 76)
```

In [57]:
```
t[2]=10
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Input In [57], in <cell line: 1>()
----> 1 t[2]=10

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

set

```
In [58]: s={10,56,67,264,264,10,56,23,34,12,21,23}
```

```
In [63]: type(s)
```

```
Out[63]: set
```

```
In [59]: s
```

```
Out[59]: {10, 12, 21, 23, 34, 56, 67, 264}
```

dictionary

```
In [76]: d={"name":"vaishnavi","age":22,"sex":"female"}
```

```
In [77]: type(d)
```

```
Out[77]: dict
```

```
In [78]: d
```

```
Out[78]: {'name': 'vaishnavi', 'age': 22, 'sex': 'female'}
```

```
In [79]: d[0]
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [79], in <cell line: 1>()
----> 1 d[0]

KeyError: 0
```

```
In [81]: #we cannot call it through index we have to use its respective key to call
         d["name"]
```

```
Out[81]: 'vaishnavi'
```

Arithmetic operations

```
In [121]: a=10
          b=20
```

```
In [122]: result=a+b
```

In [123]:
```
print(result)
```
```
30
```

In [124]:
```
result=a-b
```

In [125]:
```
print(result)
```
```
-10
```

In [126]:
```
result=a*b
```

In [127]:
```
print(result)
```
```
200
```

In [128]:
```
result=a/b
```

In [129]:
```
print(result)
```
```
0.5
```

In [130]:
```
#if you want your result in integer but not in float
result=a//b
```

In [131]:
```
print(result)
```
```
0
```

In [132]:
```
result=a%b
```

In [133]:
```
print(result)
```
```
10
```

In [136]:
```
a=10.5
b=20.5
```

In [137]:
```
result=x/y
```

In [138]:
```
print(result)
```
```
9.803921568627452
```

String operations

In [139]:
```
s="vaishnavi abbugari"
```

```
In [140]: s[0]
Out[140]: 'v'

In [146]: s[4:]
Out[146]: 'hnavi abbugari'

In [145]: s[:4]
Out[145]: 'vais'

In [147]: s[0:6]
Out[147]: 'vaishn'

In [148]: s[1:-1]
Out[148]: 'aishnavi abbugar'

In [149]: s[::-1]
Out[149]: 'iragubba ivanhsiav'

In [151]: s[0:50]
Out[151]: 'vaishnavi abbugari'

In [152]: len(s)
Out[152]: 18
```

Complex numbers

```
In [153]: num=29+7j

In [156]: type(num)
Out[156]: complex

In [157]: num.real
Out[157]: 29.0

In [159]: num.imag
Out[159]: 7.0
```

Conversions

```
In [160]: x=100
```

```
In [161]: type(x)
```

```
Out[161]: int
```

```
In [162]: x="100"
```

```
In [163]: type(x)
```

```
Out[163]: str
```

```
In [164]: #to convert str into int
          int(x)
```

```
Out[164]: 100
```

```
In [167]: #to convert str into int permanently
          x=int(x)
```

```
In [169]: type(x)
```

```
Out[169]: int
```

```
In [170]: x=float(x)
```

```
In [171]: type(x)
```

```
Out[171]: float
```

```
In [172]: x=complex(x)
```

```
In [173]: type(x)
```

```
Out[173]: complex
```

```
In [174]: x
```

```
Out[174]: (100+0j)
```

Functions in numbers

```
In [178]: x=_5.6
```

```
In [180]: #if your number is positive or negative the output will always comes as posi
          abs(x)
```

```
Out[180]: 5.6
```

In [184]:
```
import math
x=10
```

In [186]:
```
math.exp(x)
```

Out[186]:
```
22026.465794806718
```

In [187]:
```
math.e
```

Out[187]:
```
2.718281828459045
```

In [189]:
```
math.pi
```

Out[189]:
```
3.141592653589793
```

In [190]:
```
math.sqrt(9)
```

Out[190]:
```
3.0
```

In [191]:
```
max(100,267,5879742,6547,824678)
```

Out[191]:
```
5879742
```

In [192]:
```
min(645,6348914,64575,75)
```

Out[192]:
```
75
```

String methods

In [228]:
```
s="Hello World"
```

In [229]:
```
#it will capitalize the first letter
s.capitalize()
```

Out[229]:
```
'Hello world'
```

In [231]:
```
#it will lower the characters
s.lower()
```

Out[231]:
```
'hello world'
```

In [233]:
```
#it will upper the characters
s.upper()
```

Out[233]:
```
'HELLO WORLD'
```

In [244]:
```
#it will return a centered string of length width
s.center(20,"*")
```

Out[244]:
```
'****Hello World*****'
```

In [252]:
```
#it will count how many times the character is repeated
s.count("l")
```
Out[252]:
```
3
```

In [270]:
```
#it will give the index of the character
s.index("l")
```
Out[270]:
```
2
```

In [274]:
```
#it will give the index of a word
s.find("rl")
```
Out[274]:
```
8
```

In [278]:
```
#it will replace the value with new one
s.replace("W","@")
```
Out[278]:
```
'Hello @orld'
```

In [282]:
```
#it will split the values by character
s.split("o")
```
Out[282]:
```
['Hell', ' W', 'rld']
```

In [292]:
```
s1="hello123"
```

In [294]:
```
#does it contains alphabets and numneric?
s1.isalnum()
```
Out[294]:
```
True
```

In [296]:
```
#does it contains all numbers?
s1.isnumeric()
```
Out[296]:
```
False
```

In [298]:
```
#it will say weather the characters are upper or not
s.isupper()
```
Out[298]:
```
False
```

In [300]:
```
#it will say weather the characters lower or not
s.islower()
```
Out[300]:
```
False
```

In [306]:
```
#to change the original value
s
```
Out[306]:
```
'Hello hawaiii'
```

```
In [302]: s.replace("World","hawaiii")
Out[302]: 'Hello hawaiii'

In [303]: s
Out[303]: 'Hello World'

In [304]: s=s.replace("World","hawaiii")

In [305]: s
Out[305]: 'Hello hawaiii'
```

Lists

```
In [307]: l=[10,"vaishnavi",10.6,10.10j]

In [308]: l
Out[308]: [10, 'vaishnavi', 10.6, 10.1j]

In [309]: l[1]
Out[309]: 'vaishnavi'

In [311]: mat=[[1,2],[3,4]]

In [312]: mat
Out[312]: [[1, 2], [3, 4]]

In [313]: mat[0]
Out[313]: [1, 2]

In [314]: mat[1]
Out[314]: [3, 4]
```

operations in lists

```
In [316]: z=[0]*100
```

In [317]:
```
z
```

Out[317]:
```
[0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0,
 0]
```

In [318]:

```python
a=["vaishmnavi"]
b=["abbugari"]
a+b
```

Out[318]:

```
['vaishmnavi', 'abbugari']
```

In [319]:
```
list("hey there")
```
Out[319]:
```
['h', 'e', 'y', ' ', 't', 'h', 'e', 'r', 'e']
```

In [417]:
```
num=[1,2,3,4,5]
first,*other=num
print(first)
print(other)
```
```
1
[2, 3, 4, 5]
```

list methods

In [329]:
```
l
```
Out[329]:
```
[10, 'vaishnavi', 10.6, 10.1j]
```

In [338]:
```
#appends a new character at the end
l.append(20)
```

In [339]:
```
l
```
Out[339]:
```
[10, 'vaishnavi', 10.6, 10.1j, 100, 100, 100, 20, 20]
```

In [342]:
```
l.extend(s)
```

In [344]:
```
#appends the whole variable
l
```

Out[344]:
```
[10,
 'vaishnavi',
 10.6,
 10.1j,
 100,
 100,
 100,
 20,
 20,
 'name',
 'age',
 'sex',
 'H',
 'e',
 'l',
 'l',
 'o',
 ' ',
 'h',
 'a',
 'w',
 'a',
 'i',
 'i',
 'i']
```

In [348]:
```
#appends the character at particular value
l.insert(2,"hahaha")
```

In [349]:
```
l
```
Out[349]:
```
[10,
 'vaishnavi',
 'hahaha',
 'hahaha',
 10.6,
 10.1j,
 100,
 100,
 100,
 20,
 20,
 'name',
 'age',
 'sex',
 'H',
 'e',
 'l',
 'l',
 'o',
 ' ',
 'h',
 'a',
 'w',
 'a',
 'i',
 'i',
 'i']
```

In [353]:
```
#counts the number of repetations of a character
l.count("i")
```
Out[353]:
```
3
```

In [384]:
```
l=[1,2,3]
```

In [385]:
```
#clears the all characters in list
l.clear()
```

In [386]:
```
l
```
Out[386]:
```
[]
```

In [387]:
```
l=[12,23,34,56]
```

In [388]:
```
#gives the index of the character
l.index(23)
```
Out[388]:
```
1
```

In [389]:
```
#removes particular character
l.remove(56)
```

In [390]:
```
l
```
Out[390]:
```
[12, 23, 34]
```

In [391]:
```
#removes last character in list
l.pop()
```
Out[391]:
```
34
```

In [393]:
```
#removes selected character using index
l.pop(0)
```
Out[393]:
```
12
```

In [394]:
```
l
```
Out[394]:
```
[23]
```

In [396]:
```
l=[12,23,34,56]
```

In [399]:
```
#reverse the entire list
l.reverse()
```

In [398]:
```
l
```
Out[398]:
```
[56, 34, 23, 12]
```

In [403]:
```
#sorting in ascending order
l.sort()
```

In [404]:
```
l
```
Out[404]:
```
[12, 23, 34, 56]
```

In [409]:
```
#sorting in descing order
l.sort(reverse=True)
```

In [410]:
```
l
```
Out[410]:
```
[56, 34, 23, 12]
```

In [413]:
```
#returns the copy
l.copy()
```
Out[413]:
```
[56, 34, 23, 12]
```

In [414]:
```
l
```
Out[414]:
```
[56, 34, 23, 12]
```

built-in functions in list

In [418]:
```
min(l)
```
Out[418]:
```
12
```

In [419]:
```
max(l)
```
Out[419]:
```
56
```

In [421]:
```
sum(l)
```
Out[421]:
```
125
```

In [422]:
```
len(l)
```
Out[422]:
```
4
```

In [425]:
```
#average
sum(l)/len(l)
```
Out[425]:
```
31.25
```

Tuples

In [427]:
```
t=()
```

In [428]:
```
type(t)
```
Out[428]:
```
tuple
```

In [429]:
```
t
```
Out[429]:
```
()
```

In [439]:
```
t=(12,23,34,56)
```

In [440]:
```
t
```
Out[440]:
```
(12, 23, 34, 56)
```

In [441]:
```
cities="pune","hyderabad","mumbai","chennai"
```

In [442]:
```
cities
```
Out[442]:
```
('pune', 'hyderabad', 'mumbai', 'chennai')
```

In [443]:
```
#deleting tuple
del(t)
```

```
In [444]: t

---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Input In [444], in <cell line: 1>()
----> 1 t

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

```
In [460]: #converting list into tuple
          l
Out[460]: [56, 34, 23, 12]
```

```
In [475]: type(l)
Out[475]: list
```

```
In [476]: t
Out[476]: (12, 23, 34, 56)
```

```
In [477]: type(t)
Out[477]: tuple
```

```
In [478]: l_to_t=tuple(l)
```

```
In [479]: l_to_t
Out[479]: (56, 34, 23, 12)
```

```
In [488]: type(l_to_t)
Out[488]: tuple
```

nested tuples in a list

```
In [489]: list=[(1,2,3),(4,5,6)]
```

```
In [490]: list
Out[490]: [(1, 2, 3), (4, 5, 6)]
```

```
In [491]: list.append(1)
```

```
In [492]: list
Out[492]: [(1, 2, 3), (4, 5, 6), 1]
```

In [493]:
```
list.pop()
```
Out[493]:
```
1
```

In [494]:
```
list
```
Out[494]:
```
[(1, 2, 3), (4, 5, 6)]
```

nested list in tuples

In [495]:
```
tuple=([1,2,3],[45,6,])
```

In [501]:
```
tuple.append(78)
```
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [501], in <cell line: 1>()
----> 1 tuple.append(78)

AttributeError: 'tuple' object has no attribute 'append'
```

In [502]:
```
tuple[0].append(78)
```

In [503]:
```
tuple
```
Out[503]:
```
([1, 2, 3, 78, 78], [45, 6])
```

Dictionaries

In [504]:
```
d={}
```

In [505]:
```
type(d)
```
Out[505]:
```
dict
```

In [510]:
```
d1={"name":"vaishnavi","age":22,"sex":"female","course":"fsds"}
```

In [511]:
```
d1
```
Out[511]:
```
{'name': 'vaishnavi', 'age': 22, 'sex': 'female', 'course': 'fsds'}
```

In [514]:
```
d1["name"]
```
Out[514]:
```
'vaishnavi'
```

In [518]:
```
d1["location"]="hyderabad"
```

In [519]:
```
d1
```
Out[519]:
```
{'name': 'vaishnavi',
 'age': 22,
 'sex': 'female',
 'course': 'fsds',
 'location': 'hyderabad'}
```

In [530]:
```
d1["skills"]={"python":"basics","sql":"advanced","excel":"basic"}
```

In [531]:
```
d1
```
Out[531]:
```
{'name': 'vaishnavi',
 'age': 22,
 'sex': 'female',
 'course': 'fsds',
 'location': 'hyderabad',
 'skills': {'python': 'basics', 'sql': 'advanced', 'excel': 'basic'}}
```

In [534]:
```
d1["skills"]["python"]
```
Out[534]:
```
'basics'
```

In [544]:
```
d1
```
Out[544]:
```
{'name': 'vaishnavi',
 'age': 22,
 'sex': 'female',
 'course': 'fsds',
 'location': 'hyderabad'}
```

dictonary methods

In [548]:
```
d1.fromkeys("name")
```
Out[548]:
```
{'n': None, 'a': None, 'm': None, 'e': None}
```

In [549]:
```
d1.items()
```
Out[549]:
```
dict_items([('name', 'vaishnavi'), ('age', 22), ('sex', 'female'), ('course',
'fsds'), ('location', 'hyderabad')])
```

In [550]:
```
print(d1.get("course"))
```
```
fsds
```

In [551]:
```
d1.popitem()
```
Out[551]:
```
('location', 'hyderabad')
```

In [552]:
```
d1.values()
```
Out[552]:
```
dict_values(['vaishnavi', 22, 'female', 'fsds'])
```

In [554]:
```
keys={"a","b","c","d"}
values=1
dict.fromkeys(keys,values)
```
Out[554]:
```
{'d': 1, 'c': 1, 'b': 1, 'a': 1}
```

Sets

In [557]:
```
s={10,2,0,"vaishnavi",0,0,0,20,20,30,}
```

In [558]:
```
s
```
Out[558]:
```
{0, 10, 2, 20, 30, 'vaishnavi'}
```

Methods in sets

In [560]:
```
s.add("s")
```

In [561]:
```
s
```
Out[561]:
```
{0, 10, 2, 20, 30, 's', 'vaishnavi'}
```

In [562]:
```
fs=frozenset([12,34,24])
```

In [563]:
```
fs
```
Out[563]:
```
frozenset({12, 24, 34})
```

In [565]:
```
fs.add(2)
```
```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Input In [565], in <cell line: 1>()
----> 1 fs.add(2)

AttributeError: 'frozenset' object has no attribute 'add'
```

In [572]:
```
#discards the given character
s.discard("s")
```

In [573]:
```
s
```
Out[573]:
```
{10, 2, 20, 30, 'vaishnavi'}
```

In [574]:
```
s.remove(2)
```

In [575]:
```
s
```

Out[575]:
```
{10, 20, 30, 'vaishnavi'}
```

In [579]:
```
#difference between discard and remove is discard doesnt give any error if t
#but remove gives an key error if the given argument is not in the list
```

In [580]:
```
s.discard("hahaha")
```

In [581]:
```
s.remove("hahaha")
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
Input In [581], in <cell line: 1>()
----> 1 s.remove("hahaha")

KeyError: 'hahaha'
```

In [582]:
```
s.pop()
```

Out[582]:
```
'vaishnavi'
```

In [583]:
```
s
```

Out[583]:
```
{10, 20, 30}
```

In [585]:
```
s1={10,20,30,30,40}
s2={40,50,60,70,80}
```

In [587]:
```
#adds 2 sets and gives in 1 set
s1.union(s2)
```

Out[587]:
```
{10, 20, 30, 40, 50, 60, 70, 80}
```

In [590]:
```
#updates the set with union values
s1.update(s2)
```

In [591]:
```
s1
```

Out[591]:
```
{10, 20, 30, 40, 50, 60, 70, 80}
```

In [595]:
```
#gives the comman values in 2sets
s1.intersection(s2)
```

Out[595]:
```
{40, 50, 60, 70, 80}
```

In [599]:
```
#it permanently updates the intersection values
s1.intersection_update(s2)
```

In [601]:
```
#comman values
s1
```
Out[601]:
```
{40, 50, 60, 70, 80}
```

In [603]:
```
s1={10,20,30,30,40}
s2={40,50,60,70,80}
```

In [605]:
```
#gives uncomman values of 2sets
s1.difference(s2)
```
Out[605]:
```
{10, 20, 30}
```

In [609]:
```
##it permanently updates the differnt values
s1.difference_update(s2)
```

In [607]:
```
#uncommon values
s1
```
Out[607]:
```
{10, 20, 30}
```

In [615]:
```
s1={10,20,30,40,50,60,70,30}
s2={50,60,70,30}
```

In [619]:
```
#does all the elements in s1 are avaialable in s2?
s1.issubset(s2)
```
Out[619]:
```
False
```

In [622]:
```
#does all the elements in s2 are avaialable in s2?
s2.issubset(s1)
```
Out[622]:
```
True
```

In [623]:
```
s1.issuperset(s2)
```
Out[623]:
```
True
```

In [625]:
```
s2.issuperset(s1)
```
Out[625]:
```
False
```

In [628]:
```
s1={10,20,30}
s2={40,50,60}
```

In [630]:
```
#no interactions between two sets
s1.isdisjoint(s2)
```
Out[630]:
```
True
```