In [1]:
```
#checking the verion
import sys
sys.version
```

Out[1]:
```
'3.12.4 | packaged by Anaconda, Inc. | (main, Jun 18 2024, 15:03:56) [MSC v.1929 6
4 bit (AMD64)]'
```

In [3]:
```
#with writing of print statement we can get the statements that we need
a=3
b=4
print(a)
print(b)
```

```
3
4
```

In [5]:
```
#syntax of the variable is variable=value,if we give value=variable then it shows t
a=3
print(a)
#here a is the variable
```

```
3
```

sum of 2 num

In [8]:
```
#sum of two numbers
num1=3
num2=6
sum=num1+num2
print(sum)
```

```
9
```

In [10]:
```
#if we want to the sum statements in the output then
num1=3
num2=7
sum=num1+num2
print("the sum of",num1,"and",num2,"is",sum)
```

```
the sum of 3 and 7 is 10
```

In [12]:
```
#python is case sensitive
#variable never start with number
#no special character is allowed in variable
#underscore wull return the previous value
#python keyword never be a variable
```

integers

In [15]:
```
i=4
i
```

Out[15]: 4

In [17]:
```
type(i)
```

Out[17]: int

In [19]:
```
a,b,c=4,5,6
print(a,b,c)
```

```
4 5 6
```

float

In [22]:
```
f=110.20
f
```

Out[22]: 110.2

In [ ]:

In [24]:
```
type(f)
```

Out[24]: float

In [26]:
```
f1,f2,f3=2.3,3.4,5.1
```

In [ ]:

In [28]:
```
f1=5
```

In [30]:
```
f5=2.4e2 #float data type accepts "e"
f5
```

Out[30]: 240.0

boolean

In [33]:
```
b=True
```

In [35]:
```
b1=False
```

In [37]:
```
print(b)
print(b1)
```

```
True
False
```

In [39]:
```
True+True+True+False-True
```

Out[39]: 2

In [41]:
```
True+False #true is 1 and false 0
```

Out[41]: 1

In [43]:
```
False*True
```

Out[43]: 0

In [45]:
```
True*True
```

Out[45]: 1

In [47]:
```
False/True
```

Out[47]: 0.0

complex

In [50]:
```
c=10+20J
c
```

Out[50]: (10+20j)

In [52]:
```
type(c)
```

Out[52]: complex

In [54]:
```
c.imag
```

Out[54]: 20.0

In [56]:
```
c.real
```

Out[56]: 10.0

In [58]:
```
c1=10+20j
c2=30+40j
print(c1+c2)
print(c1-c2)
```

```
(40+60j)
(-20-20j)
```

In [60]:
```
a=3+5j
a
```

Out[60]: (3+5j)

In [62]:
```
a.imag
```

Out[62]: 5.0

In [64]:
```
a.real
```

Out[64]: 3.0

strings

In [67]:
```
s='nit'
s
```

Out[67]: 'nit'

In [69]:
```
type(s)
```

Out[69]: str

In [71]:
```
s1="hello python"
s1
```

Out[71]: 'hello python'

In [73]:
```
s2='''nit python'''
s2
```

Out[73]: 'nit python'

string indexing

In [76]:
```
s1[0]
```

Out[76]: 'h'

s1[-4]

In [78]:
```
s1[4]
```

Out[78]: 'o'

In [80]:
```
s1[-4]
```

Out[80]: 't'

In [82]:
```
s1[-7]
```

Out[82]: ' '

In [84]:
```
s
```

Out[84]: 'nit'

In [86]:
```
'nit'
```

Out[86]: 'nit'

In [88]:
```
'hello'
```

Out[88]: 'hello'

string slicing

In [91]:
```
s1[:]
```

Out[91]: 'hello python'

In [93]:
```
s1[2:7]
```

Out[93]: 'llo p'

data slicing

In [95]:
```
s3="dataanalytic"
```

In [97]:
```
s3[0:10]
```

Out[97]: 'dataanalyt'

In [99]:
```
s3[12] # when we declare the index out of range it shows error
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[99], line 1
----> 1 s3[12]

IndexError: string index out of range
```

In [101…
```
s3[9:12]
```

Out[101… 'tic'

In [103…
```
s3
```

Out[103… 'dataanalytic'

In [105…
```
s3[0:11:2]
```

Out[105… 'dtaayi'

In [107…
```
s3[2:-2]
```

Out[107…
```
'taanalyt'
```

In [109…
```
print(s)
print(s1)
print(s2)
print(s3)
```

```
nit
hello python
nit python
dataanalytic
```

type casting

other data types to int

In [111…
```
int(2.3)
```

Out[111…
```
2
```

In [113…
```
int(True) #boolen to int
```

Out[113…
```
1
```

In [115…
```
int(2+4j)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[115], line 1
----> 1 int(2+4j)

TypeError: int() argument must be a string, a bytes-like object or a real number, no
t 'complex'
```

In [117…
```
int("10")
```

Out[117…
```
10
```

In [119…
```
int("nit")
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[119], line 1
----> 1 int("nit")

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

In [121…
```
s1=("hello")
s2=("nit")
```

```
s3=(s1+s2)
s3
```

Out[121… 'hellonit'

In [123…
```
s=np.nan
```

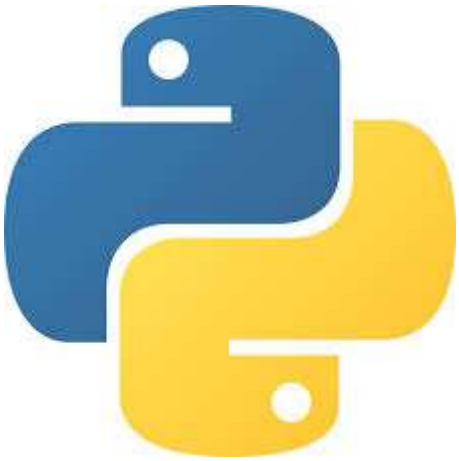
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[123], line 1
----> 1 s=np.nan

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

In [125…
```
import numpy as np
a=np.nan
```

In [127…
```
type(a)
```

Out[127… float

import the image



keywords in python

In [129…
```
import keyword
keyword.kwlist
```

Out[129…
```
['False',
 'None',
 'True',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 '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']
```

other data type to float

In [132…
```
float(3)
```

Out[132…
```
3.0
```

In [134…
```
float(True)
```

Out[134…
```
1.0
```

In [136…
```
fl0at(1+2j)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[136], line 1
----> 1 fl0at(1+2j)

NameError: name 'fl0at' is not defined
```

In [138…
```
float(3,4) # float accepts only one argument
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[138], line 1
----> 1 float(3,4)

TypeError: float expected at most 1 argument, got 2
```

In [140…
```
float("10")
```

Out[140… 10.0

other data type to complex

In [142…
```
complex(10)
```

Out[142… (10+0j)

In [144…
```
complex(10,20)
```

Out[144… (10+20j)

In [146…
```
complex(2.3,10)
```

Out[146… (2.3+10j)

In [148…
```
complex(True)
```

Out[148… (1+0j)

In [150…
```
complex(False)
```

Out[150… 0j

In [152…
```
complex("10")
```

Out[152… (10+0j)

other data types to bool

In [155…
```
bool(1)
```

Out[155… True

In [157…
```
bool(0)
```

Out[157… False

In [159…
```
bool(2.3)
```

Out[159… True

In [161…
```
bool()
```

Out[161… False

In [163…
```
bool( )
```

Out[163… False

In [165…
```
bool("nit")
```

Out[165… True

In [167…
```
bool(10+2j)
```

Out[167… True

In [169…
```
bool(0+0j)
```

Out[169… False

other datatypes to string

In [172…
```
print(str(2))
print(str(2.3))
print(str(True))
print(str(1+2j))
```

```
2
2.3
True
(1+2j)
```

indexing and slicing

In [175…
```
index="hellopython"
index
```

Out[175… 'hellopython'

In [177…
```
index[::]
```

Out[177…
```
'hellopython'
```

In [179…
```
index[:: -1] #reverse the string
```

Out[179…
```
'nohtypolleh'
```

In [181…
```
index[::-2]
```

Out[181…
```
'nhyolh'
```

In [183…
```
index[::-4]
```

Out[183…
```
'nyl'
```

In [185…
```
index[:-4]
```

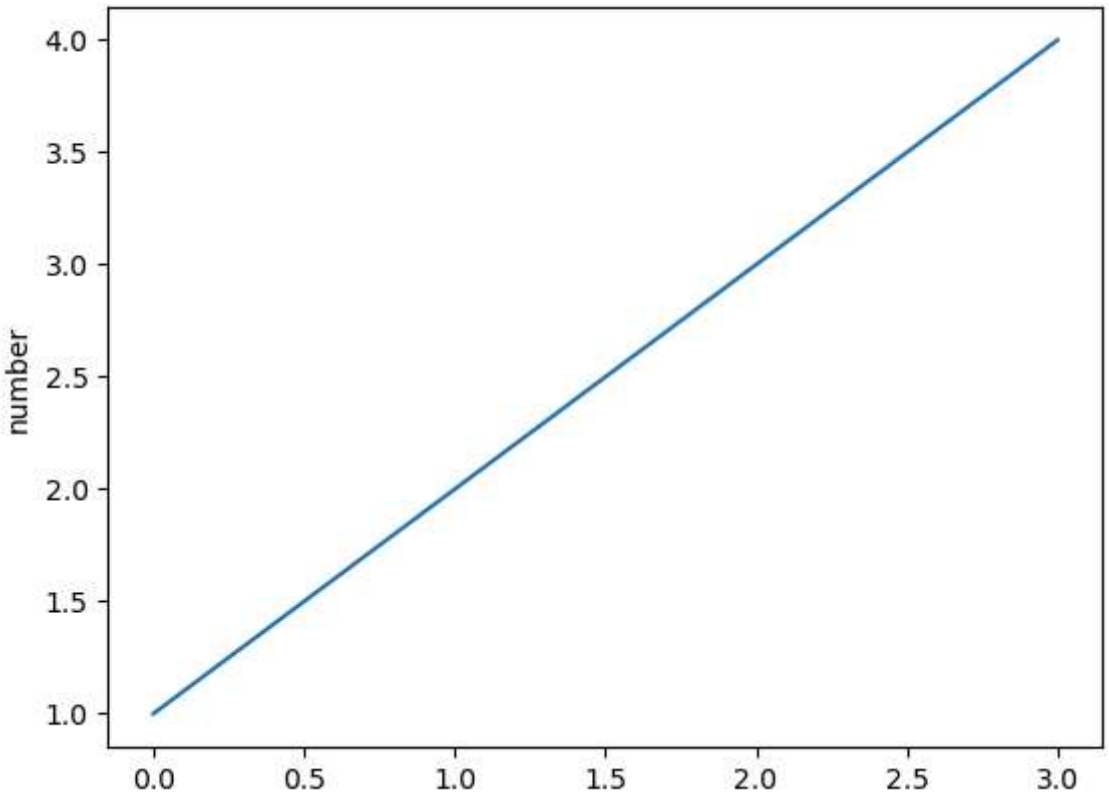
Out[185…
```
'hellopy'
```

In [187…
```
index[1:10:3]
```

Out[187…
```
'eot'
```

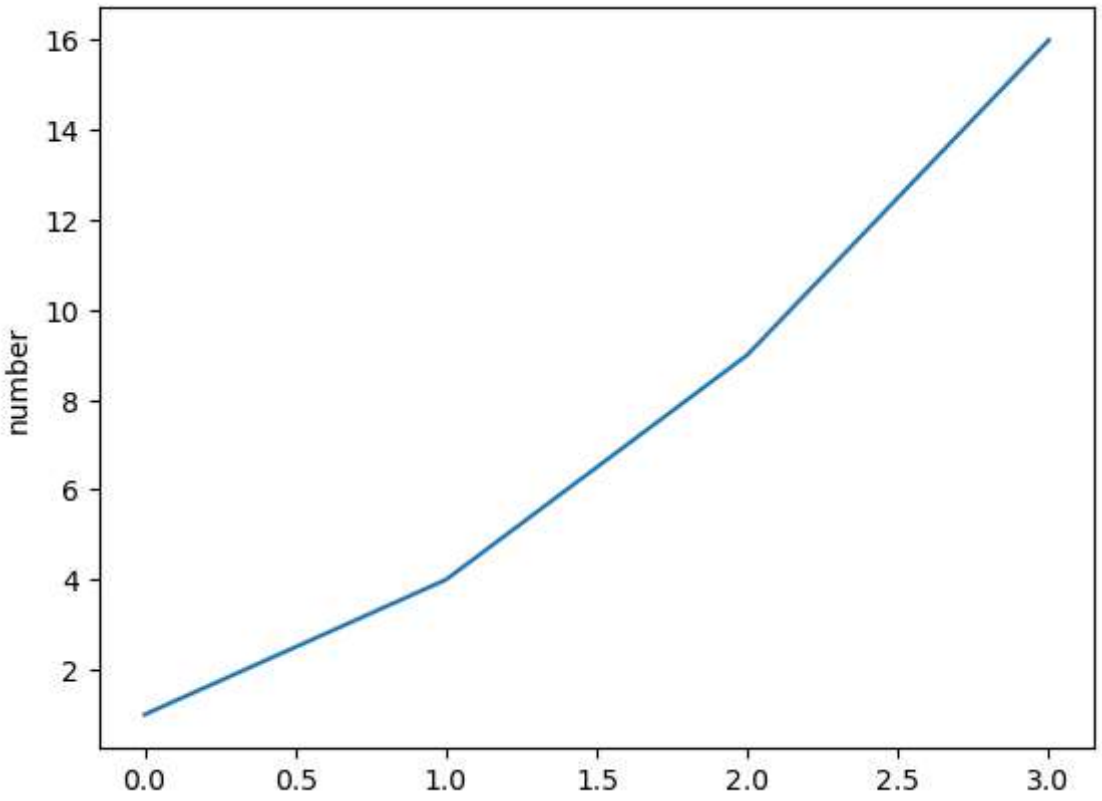
ploting the graph

In [190…
```
import matplotlib.pyplot as plt
plt.plot([1,2,3,4])
plt.ylabel('number')
plt.show()
```



In [191…

```
import matplotlib.pyplot as plt
plt.plot([1,4,9,16])
plt.ylabel('number')
plt.show()
```



adding of 2 numbers can be printed in 3 different ways

In [195…
```
num1=4
num2=5
add=num1+num2
print("addition of ",num1,"and",num2,"is=",add)
print("addition of the {} and {} is={}".format(num1,num2,add))
print(f"addition of the {num1} and {num2} is={add}")
```

```
addition of  4 and 5 is= 9
addition of the 4 and 5 is=9
addition of the 4 and 5 is=9
```

end statement

-if we want to attach the code in second line at the end of the first line there we use this ststement

In [199…
```
print("hello", end=" ")
print("how r u")
```

```
hello how r u
```

separator

- when we wnt to separate the the terms we use it
- we can use special characters to separate the terms)

In [203…
```
print("hii","hello","how r u",sep=" @ ")
```

```
hii @ hello @ how r u
```

In [205…
```
print("hii","hello","how r u",sep= " $ ")
```

```
hii $ hello $ how r u
```

average of the numbers

In [208…
```
num1=3
num2=4
average=num1+num2/2
average
```

Out[208… 5.0

In [210…
```
num1=3
num2=4
num3=5
average=(num1+num2+num3/3)
average1=(num1+num2+num3/3,2)
print( " the average of {},{}and{} is={}".format(num1,num2,num3,average or average1
```

```
the average of 3,4and5 is=8.666666666666666
```

string functions

In [213…
```
s="sushmaramchander"
s.upper()
```

Out[213… 'SUSHMARAMCHANDER'

In [215…
```
s="Sushmaramchander"
s.capitalize()
```

Out[215… 'Sushmaramchander'

In [217…
```
   print(s.lower())
print(s.title())
print(s.swapcase())
```

```
sushmaramchander
Sushmaramchander
sUSHMARAMCHANDER
```

data structures

- list
- tuple
- dict
- set

list

-denote with l -doesn't allow

In [223…
```
l=[]
l
```

Out[223… []

In [225…
```
type(l)
```

Out[225… list

append function in the list

In [228…
```
l.append(10)
l
```

Out[228… [10]

In [230…
```
l.append(20)
l.append(30)
l
```

Out[230… [10, 20, 30]

In [232…
```
l.append(2.3)
l
```

Out[232… [10, 20, 30, 2.3]

In [234…
```
l.append(1+2j)
l.append("nit")
l.append(True)
l
```

Out[234… [10, 20, 30, 2.3, (1+2j), 'nit', True]

length

In [237…
```
len(l)
```

Out[237… 7

copy function

In [240…
```
l1=l.copy()
l1
```

Out[240… [10, 20, 30, 2.3, (1+2j), 'nit', True]

In [242…
```
l==l1
```

Out[242… True

count

In [245…
```
l.count(20)
```
Out[245… 1

In [247…
```
l
```
Out[247… [10, 20, 30, 2.3, (1+2j), 'nit', True]

In [249…
```
l[:]
```
Out[249… [10, 20, 30, 2.3, (1+2j), 'nit', True]

In [251…
```
l[4]
```
Out[251… (1+2j)

In [254…
```
l2=[]
l2
```
Out[254… []

In [256…
```
l2.append(1)
l2.append(2.3)
l2.append(True)
l2.append(1+2j)
l2.append("nit")
l2
```
Out[256… [1, 2.3, True, (1+2j), 'nit']

In [258…
```
len(l2)
```
Out[258… 5

clear

In [261…
```
l2.clear
l2
```
Out[261… [1, 2.3, True, (1+2j), 'nit']

In [263…
```
len(l2)
```
Out[263… 5

In [265…
```
l2.append(1)
l2.append(2.3)
l2.append(True)
l2.append(1+2j)
```

```
l2.append("nit")
l2
```

Out[265… [1, 2.3, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

remove

In [268…
```
l2.remove(2.3)
l2
```

Out[268… [1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

In [270…
```
l3=[]
l3.append(10)
l3
```

Out[270… [10]

In [272…
```
l2
```

Out[272… [1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

extend

In [275…
```
l3.extend(l2)
l3
```

Out[275… [10, 1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

index

In [278…
```
l3.index(1+2j)
```

Out[278… 3

In [280…
```
l2.index("nit")
```

Out[280… 3

In [282…
```
l3
```

Out[282… [10, 1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

In [284…
```
l2
```

Out[284… [1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']

insert

In [287…
```
l2.insert(4,"technology")
l2
```

Out[287…
```
[1, True, (1+2j), 'nit', 'technology', 1, 2.3, True, (1+2j), 'nit']
```

In [289…
```
l2.insert(3,False)
l2
```

Out[289…
```
[1, True, (1+2j), False, 'nit', 'technology', 1, 2.3, True, (1+2j), 'nit']
```

In [291…
```
l3
```

Out[291…
```
[10, 1, True, (1+2j), 'nit', 1, 2.3, True, (1+2j), 'nit']
```

pop

In [294…
```
l3.pop()
```

Out[294…
```
'nit'
```

In [296…
```
l3.pop(4)
```

Out[296…
```
'nit'
```

In [298…
```
l3.pop(1)
```

Out[298…
```
1
```

In [300…
```
l4=[10,100,3,45,76,24]
l4
```

Out[300…
```
[10, 100, 3, 45, 76, 24]
```

sort

In [303…
```
l4.sort()
```

In [305…
```
l4
```

Out[305…
```
[3, 10, 24, 45, 76, 100]
```

In [307…
```
l4.sort(reverse=True)
l4
```

Out[307… [100, 76, 45, 24, 10, 3]

In [309…
```
l5=["z","m","c","w"]
l5
```

Out[309… ['z', 'm', 'c', 'w']

In [311…
```
l5.sort()
l5
```

Out[311… ['c', 'm', 'w', 'z']

In [313…
```
l3
```

Out[313… [10, True, (1+2j), 1, 2.3, True, (1+2j)]

reverse

In [316…
```
l3.reverse()
l3
```

Out[316… [(1+2j), True, 2.3, 1, (1+2j), True, 10]

In [318…
```
l2[3]
```

Out[318… False

In [320…
```
l2=["hii",4,False]
```

In [322…
```
l2
```

Out[322… ['hii', 4, False]

In [324…
```
l4=["nit",False,2] # mutable:-we cqn modify or change ,remove the elements in the l

l4
```

Out[324… ['nit', False, 2]

In [326…
```
l4[1]=0
```

In [328…
```
l4
```

Out[328… ['nit', 0, 2]

In [330…
```
for i in l4:
    print(i)
```

```
nit
0
2
```

In [332…
```
l6=["sbi","icic"]
l7=["hdfs","kotak"]
family_bank=l6+l7
family_bank
```

Out[332…
```
['sbi', 'icic', 'hdfs', 'kotak']
```

In [334…
```
for i in enumerate (l4):
    print(i)
```

```
(0, 'nit')
(1, 0)
(2, 2)
```

tuple

- assigned vth "( )"
- immutable

In [338…
```
t=()
t
```

Out[338…
```
()
```

In [340…
```
type(t)
```

Out[340…
```
tuple
```

In [342…
```
t1=tuple()
type(t1)
```

Out[342…
```
tuple
```

In [344…
```
t=(10,10,20,30)
t
```

Out[344…
```
(10, 10, 20, 30)
```

In [346…
```
t1=(True,1.2,"nit",3,1+2j)
t1
```

Out[346…
```
(True, 1.2, 'nit', 3, (1+2j))
```

count

In [349…
```
t1.count(10)
```

Out[349…
```
0
```

In [351…
```
t1.index("nit")
```

Out[351…
```
2
```

for loop

In [354…
```
for i in t1:
    print(i)
```

```
True
1.2
nit
3
(1+2j)
```

In [356…
```
t[:]
```

Out[356…
```
(10, 10, 20, 30)
```

In [358…
```
t4=t*4
t4
```

Out[358…
```
(10, 10, 20, 30, 10, 10, 20, 30, 10, 10, 20, 30, 10, 10, 20, 30)
```

set

- set and dict both are represented with"{ }"
- if we need to create the set then we need to use "set()". so system take it as set ot herwise by defsult it consider it as dict
- in set indexing and slicing is not acceptable

In [362…
```
s={}
s
```

Out[362…
```
{}
```

In [364…
```
type(s)
```

Out[364…
```
dict
```

In [366…
```
s1=set()
type(s1)
```

Out[366… set

In [368…
```
s1={100,20,3,15,47} # if same data type is used in the set then it automatically ar
s1
```

Out[368… {3, 15, 20, 47, 100}

In [370…
```
s2={2.3,4.5,1.3}
s2
```

Out[370… {1.3, 2.3, 4.5}

In [372…
```
s3={"z","m","a","x"}
s3
```

Out[372… {'a', 'm', 'x', 'z'}

In [374…
```
s4={10,2.3,"a",5,True,6,7} #when set contains mixed data types in it then the arran
s4
```

Out[374… {10, 2.3, 5, 6, 7, True, 'a'}

In [376…
```
print(s1)
print(s2)
print(s3)
print(s4)
```

```
{3, 100, 20, 47, 15}
{1.3, 2.3, 4.5}
{'a', 'x', 'm', 'z'}
{True, 2.3, 5, 6, 7, 'a', 10}
```

In [378…
```
for i in s1:
    print(i)
```

```
3
100
20
47
15
```

add function in set

In [381…
```
s4.add(10)
s4.add(20)
s4.add(2.3)
```

In [383…
```
s4
```

Out[383… {10, 2.3, 20, 5, 6, 7, True, 'a'}

In [385…
```
s1.add(4)
s1
```

Out[385…
```
{3, 4, 15, 20, 47, 100}
```

In [387…
```
len(s4)
```

Out[387…
```
8
```

clear

In [390…
```
s4.clear()
```

In [392…
```
s4
```

Out[392…
```
set()
```

In [394…
```
len(s4)
```

Out[394…
```
0
```

In [396…
```
del s4
```

In [398…
```
s4
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[398], line 1
----> 1 s4

NameError: name 's4' is not defined
```

copy

In [401…
```
s4=s1.copy()
s4
```

Out[401…
```
{3, 4, 15, 20, 47, 100}
```

In [403…
```
s1==s4
```

Out[403…
```
True
```

remove

In [406…
```
s1.remove(100)
s1
```

Out[406… {3, 4, 15, 20, 47}

pop

In [409…
```
s1.pop() # in set pop remove a random irrespective of order
```

Out[409… 3

In [411…
```
s2.pop()
```

Out[411… 1.3

In [413…
```
s3
```

Out[413… {'a', 'm', 'x', 'z'}

In [415…
```
"a" in s3
```

Out[415… True

set operation

In [418…
```
A={1,2,3,4,5}
B={4,5,6,7,8}
c={8,9,10}
```

union

In [421…
```
A.union(B)
```

Out[421… {1, 2, 3, 4, 5, 6, 7, 8}

In [423…
```
d_union=A.union(B)
d_union
```

Out[423… {1, 2, 3, 4, 5, 6, 7, 8}

In [425…
```
print(A)
print(B)
print(c)
print(d_union)
```

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

In [427…
```
B.union(A,c)
```

Out[427…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [429…
```
A|B
```

Out[429…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

update

In [432…
```
A.update(B)
```

In [434…
```
print(A)
print(B)
print(c)
print(d_union)
```

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

In [436…
```
A1={1,2,3,4,5}
B1={4,5,6,7,8}
c1={8,9,10}
```

intersection

In [439…
```
A1.intersection(B1)
```

Out[439…
```
{4, 5}
```

In [441…
```
A2={1,2,3,4,5}
B2={4,5,6,7,8}
c12={8,9,10}
```

difference

In [444…
```
A2-B2
```

Out[444…
```
{1, 2, 3}
```

In [446…
```
B2-A2
```

Out[446…
```
{6, 7, 8}
```

In [448…
```
B2.difference(A2)
```

Out[448… {6, 7, 8}

symmetric difference

In [451…
```
B2.symmetric_difference(A2)
```

Out[451… {1, 2, 3, 6, 7, 8}

In [453…
```
s={2,3,4,5}
```

In [455…
```
for i in s:
    print(i)
```

```
2
3
4
5
```

In [457…
```
for i in enumerate( s):
    print(i)
```

```
(0, 2)
(1, 3)
(2, 4)
(3, 5)
```

In [459…
```
a5={1,2,3,4,5,6,7,8,9}
b5={3,4,5,6,7,8}
c5={10,20,30,40}
```

In [461…
```
a5.issuperset(b5)
```

Out[461… True

In [463…
```
a5.issubset(b5)
```

Out[463… False

In [465…
```
b5.issubset(a5)
```

Out[465… True

In [467…
```
c5.isdisjoint(a5)
```

Out[467… True

In [469…
```
c5.issubset(a5)
```

Out[469… False

In [471…
```
a5
```

Out[471… {1, 2, 3, 4, 5, 6, 7, 8, 9}

In [473…
```
sum(a5)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[473], line 1
----> 1 sum(a5)

TypeError: 'int' object is not callable
```

In [475…
```
max(a5)
```

Out[475… 9

In [477…
```
min(a5)
```

Out[477… 1

In [479…
```
len(a5)
```

Out[479… 9

dict

- it is represented in key value pairs
- defined in { }
- keys can't be duplicted
- values can be duplicated

In [483…
```
d={}
type(d)
```

Out[483… dict

In [485…
```
d={1:"one",2:"two",3:"three",4:"four",5:"five"}
d
```

Out[485… {1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}

In [487…
```
d1={"one":1,"two":2,"three":3,"four":4,"five":5}
d1
```

Out[487… {'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5}

In [489…
```
print(len(d))
print(len(d1))
```

```
5
5
```

In [491…
```
d[1]    # index and slicing is not accepted
```
Out[491…
```
'one'
```

In [493…
```
d.keys()
```
Out[493…
```
dict_keys([1, 2, 3, 4, 5])
```

In [495…
```
d.values()
```
Out[495…
```
dict_values(['one', 'two', 'three', 'four', 'five'])
```

In [497…
```
 print(d.keys())
print(d.values())
```
```
dict_keys([1, 2, 3, 4, 5])
dict_values(['one', 'two', 'three', 'four', 'five'])
```

In [499…
```
d2={1:2,"2.3":4.8,"nit":"nit",True:False,1+2j:4+5j}
d2
```
Out[499…
```
{1: False, '2.3': 4.8, 'nit': 'nit', (1+2j): (4+5j)}
```

In [501…
```
d3={10:"ten",9.0:"nine"}
d3
```
Out[501…
```
{10: 'ten', 9.0: 'nine'}
```

In [503…
```
d2.items() #it gives the keys and values
```
Out[503…
```
dict_items([(1, False), ('2.3', 4.8), ('nit', 'nit'), ((1+2j), (4+5j))])
```

In [505…
```
id(d2)
```
Out[505…
```
2256229796608
```

In [507…
```
d2.pop(1)
```
Out[507…
```
False
```

In [509…
```
d2
```
Out[509…
```
{'2.3': 4.8, 'nit': 'nit', (1+2j): (4+5j)}
```

In [511…
```
d2.popitem() #random item will be deleted
```
Out[511…
```
((1+2j), (4+5j))
```

In [513…
```
d
```
Out[513…
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five'}
```

In [515…
```
for i in d:
    print(i)
```
```
1
2
3
4
5
```

In [517…
```
for i in d:
    print(i,":",d[i])
```
```
1 : one
2 : two
3 : three
4 : four
5 : five
```

dict membership

In [520…
```
d5={}
```

In [522…
```
type(d5)
```

Out[522…
```
dict
```

In [524…
```
d5={2:"hii",3:"hello",4:four}
```
```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[524], line 1
----> 1 d5={2:"hii",3:"hello",4:four}

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

In [526…
```
"hii" in d5
```

Out[526…
```
False
```

In [528…
```
range(5)
```

Out[528…
```
range(0, 5)
```

In [530…
```
r=range(5,10)
```

In [532…
```
for i in r:
    print(i)
```
```
5
6
7
8
9
```

In [534…
```
r1=range(0,11,2)
r1
```

Out[534… range(0, 11, 2)

In [536…
```
for i in r1:
    print(i)
```

```
0
2
4
6
8
10
```

In [538…
```
list(r1)
```

Out[538… [0, 2, 4, 6, 8, 10]

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: