Python Variable

In [1]:
```
import sys
sys.version
```

Out[1]:
```
'3.13.9 | packaged by Anaconda, Inc. | (main, Oct 21 2025, 19:09:58) [MSC v.192
9 64 bit (AMD64)]'
```

In [2]:
```
v = 6
v
```

Out[2]:
```
6
```

In [3]:
```
6 = v
```

```
  Cell In[3], line 1
    6 = v
    ^
SyntaxError: cannot assign to literal here. Maybe you meant '==' instead of '='?
```

In [ ]:
```
2v = 10
2v
```

In [ ]:
```
v2 = 16
v2
```

In [ ]:
```
nit = 18
NIT
```

In [ ]:
```
nit
```

In [ ]:
```
v@ = 67
v@
```

In [ ]:
```
v_ =17
v_
```

In [ ]:
```
niT_@ = 89
niT_@
```

In [ ]:
```
import keyword
keyword.kwlist
```

In [ ]:
```
if = 10
if
```

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

Python Variable Completed

In [ ]:
```
x = 16
x
```

In [ ]:
```
id(x)
```

In [ ]:
```
y = 16
y
```

In [ ]:
```
id(y)
```

Variable_name = value

In [ ]:
```
age = 56
print(age)
```

In [ ]:
```
type(age)
```

In [ ]:
```
age = 'yt'
print(age)
```

In [ ]:
```
type(age)
```

In [ ]:
```
price = 89.89
print(price)
type(price)
```

In [ ]:
```
 is_active = True
print(is_active)
```

In [ ]:
```
is_active = True
print(is_active)
```

In [ ]:
```
a=6
b=7
c= a+b
print(c)
```

In [ ]:
```
type(a)
```

In [ ]:
```
f = 2.8
f
```

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

In [ ]:
```
gold = 127898.00
gold
```

In [ ]:
```
type(gold)
```

In [ ]:
```
3e0
```

In [ ]:
```
3E1
```

In [ ]:
```
3E3
```

In [ ]:
```
4a4 # float only e or E letter is allowed that's it
```

In [ ]:
```
b = true
```

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

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

In [ ]:
```
b2 = None
b2
```

In [ ]:
```
True + True
```

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

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

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

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

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

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

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

In [ ]:
```
print(True + True)
print(True * False )
print(True - True)
```

complex data types

In [ ]:
```
c = 10 + 20j
c
```

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

In [ ]:
```
c
```

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

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

In [ ]:
```
d = 20 + 30j
d
```

In [ ]:
```
print(c)
print(d)
```

In [ ]:
```
print(c+d)
```

In [ ]:
```
c2 = 10+20T
c2
```

In [ ]:
```
welcome to nit
```

In [ ]:
```
s = 'welcom to nit'
s
```

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

In [ ]:
```
s1 = "welcom to nit"
s1
```

In [ ]:
```
s2 = '''welcom to nit'''
s2
```

In [ ]:
```
s3 = ''' welcom
        to
        nit '''
s3
```

In [ ]:
```
s4 = ''' If you want, I can shortlist top 3 best options for your
child’s grade (like Class 6) with contact info, approximate fees,
and curriculum highlights — just tell me the grade!'''
s4
```

In [ ]:
```
s
```

In [ ]:
```
s[-1]
```

In [ ]:
```
s[-6]
```

In [ ]:
```
s[6]
```

In [ ]:
```
s[-5]
```

In [ ]:
```
print(3 + 2)   # Addition(+)
print(3 - 2)   # Subtraction(-)
print(3 * 2)   # Multiplication(*)
print(3 / 2)   # Division(/)
print(3 ** 2)  # Exponential(**)
print(3 % 2)   # Modulus(%)
print(3 // 2)  # Floor division operator(//)
```

In [ ]:
```
person_info = {
    'firstname':'Asabeneh',
    'lastname':'Yetayeh',
    'country':'Finland',
    'city':'Helsinki'
    }
person_info
```

In [ ]:
```
type(person_info)
```

In [ ]:
```
# Declaring multiple variables in one line

first_name, last_name, country, age, is_married = 'Ravi', 'kumar', 'Europe', 250
```

In [ ]:
```
print(first_name, last_name, country, age, is_married)
print('First name:', first_name)
print('Last name: ', last_name)
print('Country: ', country)
print('Age: ', age)
print('Married: ', is_married)
```

In [ ]:
```
# Variables in Python

first_name = 'RAVI'
last_name = 'KUMAR'
country = 'EUROPE'
city = 'TELENGANA'
age = 40087
is_married = False
skills = ['HTML', 'CSS', 'JS', 'React', 'Python']
person_info = {
    'firstname':'Asabeneh',
    'lastname':'Yetayeh',
    'country':'Finland',
    'city':'Helsinki'
    }

# Printing the values stored in the variables

print('First name:', first_name)
print('First name length:', len(first_name))
print('Last name: ', last_name)
print('Last name length: ', len(last_name))
print('Country: ', country)
print('City: ', city)
print('Age: ', age)
print('Married: ', is_married)
print('Skills: ', skills)
print('Person information: ', person_info)
```

Data Structure

Ravi

Ravi

In [ ]:
```
l= []
l
```

In [ ]:
```
type(l)
```

In [ ]:
```
len(l)
```

In [ ]:
```
l.append(10)
```

In [ ]:
```
l
```

In [ ]:
```
l.append(10,20,30)
l
```

In [ ]:
```
l.append(20)
l.append(30)
l.append(40)
l
```

In [ ]:
```
l1=l.copy() #l= l1

l1
```

In [ ]:
```
print(l)
print(l1)
```

In [ ]:
```
l == l1
```

In [ ]:
```
l1.append('nit')
l1.append(True)
l1.append(1+2j)
l1.append(2.3)
```

In [ ]:
```
l1
```

In [ ]:
```
l==l1
```

In [ ]:
```
print(l)
print(l1)
```

In [ ]:
```

```

In [ ]:
```
l=l1
```

In [ ]:
```

```

In [ ]:
```
l
```

In [ ]:
```
l1
```

In [ ]:
```
print(l)
print(l1)
```

In [ ]:
```
l1.clear()
```

In [ ]:
```
l1
```

In [ ]:
```
l
```

In [ ]:
```
id(l)
```

In [ ]:
```
id(l1)
```

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

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

In [ ]:
```
hi
```

In [ ]:
```
'hi'
```

In [ ]:
```
a=4
b= 'nit'
c=a+b
```

In [ ]:
```
'nit'+'nit'
```

In [ ]:
```
5*'nit'
```

In [ ]:
```
5*' nit'
```

In [ ]:
```
print('C:\nit')
```

In [ ]:
```
print(r'C:\nit')
```

In [ ]:
```
tax=20
price =100
price*tax
```

In [ ]:
```
price+_
```

In [ ]:
```
round(189.879)
```

In [ ]:
```
help()
```

In [ ]:
```
'doesn\'t'   #use  \' to escape the single quote
```

In [ ]:
```
"doesn't"
```

In [ ]:
```
'"Yes,"' they said
```

Python Data Structure

In [ ]:
```
l= []
l
```

In [ ]:
```
type(l)
```

In [ ]:
```
print(type(l))
```

In [ ]:
```
l1 =[10,20,30,40]
l1
```

In [ ]:
```
len(l1)
```

In [ ]:
```
type(l1)
```

In [ ]:
```
print(l)
print(l1)
```

In [ ]:
```
l == l1
```

In [ ]:
```
l != l1
```

In [ ]:
```
l1
```

In [ ]:
```
l
```

In [ ]:
```
l.append(100)
l.append(12.5)
l.append(True)
l.append(1+2j)
l.append([1,2,3])
```

In [ ]:
```
l
```

In [ ]:
```
100 in l
```

In [ ]:
```
1000 in l
```

In [ ]:
```
l.append(100)
```

In [ ]:
```
print(l)
print(l1)
```

In [ ]:
```
print(len(l))
print(len(l1))
```

In [ ]:
```
l.append(30,40)
```

In [ ]:
```
l.count(100)
```

In [ ]:
```
l1
```

In [ ]:
```
l.count(12.5)
```

In [ ]:
```
l.count(1000)
```

In [ ]:
```
l1
```

In [ ]:
```
l2
```

In [ ]:
```
l2 = l1.copy()
l2
```

In [ ]:
```
print(l)
print(l1)
print(l2)
```

In [ ]:
```
l1.append(50)
```

In [ ]:
```
l1
```

In [ ]:
```
l2
```

In [ ]:
```
l2.append(50)
```

In [ ]:
```
l2
```

In [ ]:
```
l2.clear()
```

In [ ]:
```
l2
```

In [ ]:
```
print(l)
print(l1)
print(l2)
```

In [ ]:
```
l2
```

In [ ]:
```
l2= l1.copy()
l2
```

In [ ]:
```
print(l)
print(l1)
print(l2)
```

In [ ]:
```
print(len(l))
print(len(l1))
print(len(l2))
```

In [ ]:
```
l
```

In [ ]:
```
l.extend(l1)
l
```

In [ ]:
```
l1
```

In [ ]:
```
l.index(10)
```

In [ ]:
```
l.index(100)
```

In [ ]:
```
l1.insert(1,15)
l1
```

In [ ]:
```
l1.insert(3,25)
l1
```

In [ ]:
```
l1
```

In [ ]:
```
l
```

In [ ]:
```
l.pop()
l
```

In [ ]:
```
l.pop()
```

In [ ]:
```
l.pop()
```

In [ ]:
```
l.pop()
```

In [ ]:
```
l.pop(4)
```

In [ ]:
```
l
```

In [ ]:
```
l.pop(3)
```

In [ ]:
```
l.remove(True)
```

In [ ]:
```
l.remove(10)
```

In [ ]:
```
l
```

In [ ]:
```
l.reverse()
l
```

In [ ]:
```
l.sort()
```

In [ ]:
```
l1
```

In [ ]:
```
l1.sort()
```

In [ ]:
```
l1
```

In [ ]:
```
l3 = ['a',2.3,1+2j,True]
```

In [ ]:
```
l3.sort()
```

In [ ]:
```
l1.sort()
```

In [ ]:
```
l1
```

In [ ]:
```
l1.sort(reverse=True)
```

In [ ]:
```
l1
```

In [ ]:
```
l1
```

In [ ]:
```
for i in l1:
    print(i)
```

In [ ]:
```
for i in enumerate(l1):
    print(i)
```

In [ ]:
```
for i in l1:print(i)
```

In [ ]:
```
l1
```

In [ ]:
```
all(l1)
```

In [ ]:
```
any(l1)
```

In [ ]:
```
l1.append(0)
l1
```

In [ ]:
```
all(l1)
```

In [ ]:
```
any(l1)
```

In [ ]:
```
l1[:]
```

In [ ]:
```
l1[3:]
```

In [ ]:
```
l1[1:10]
```

In [ ]:
```
l1[3:20]
```

In [ ]:
```
l1[2:6]
```

In [ ]:
```
l1[1:8:3]
```

In [ ]:
```
l1[::2]
```

In [ ]:
```
l1
```

In [ ]:
```
l1[::-1]
```

In [ ]:
```
l1[::-2]
```

In [ ]:
```
l[0]= 100
l
```

In [ ]:
```

```

Tuples

1. Tuple is similar to List except that the objects in tuple are immutable which means we cannot

change the elements of a tuple once assigned.

2. When we do not want to change the data over time, tuple is a preferred data type.

3. Iterating over the elements of a tuple is faster compared to iterating over a list.

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

In [38]:
```
t1 =(10,20,30)
t1
```

Out[38]: (10, 20, 30)

In [39]:
```
t1.count(10)
```

Out[39]: 1

In [40]:
```
t1.index(10)
```

Out[40]: 0

In [41]:
```
t1[0]
```

Out[41]: 10

In [42]:
```
t1[0]=100
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[42], line 1
----> 1 t1[0]=100

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

In [ ]:
```
t2 =(10,10,20)
t2
```

In [ ]:
```
t2[:]
```

In [ ]:
```
a = sorted(t2)
```

In [ ]:
```
a
```

In [ ]:
```
type(a)
```

In [43]:
```
tup1 =()
```

In [44]:
```
tup2=(10,30,60)
```

In [46]:
```
tup3 =(10.77,30.88,67.98)
```

In [47]:
```
tup4 =('one','two','three')
```

In [48]:
```
tup5 =('Ravi',67,(40,67),(345,98))
```

In [49]:
```
tup6 =(78,'Ravi','58.789')
```

In [50]:
```
tup7 =('Ravi',67,[78,87],[46,99],{'John','David'},(67,89,83))
```

In [51]:
```
len(tup7)
```

Out[51]:
```
6
```

In [ ]:
```

```

Tuple inndexing

In [52]:
```
tup2[0]
```

Out[52]:
```
10
```

In [53]:
```
tup4[0]
```

Out[53]:
```
'one'
```

In [54]:
```
tup4[0][0]
```

Out[54]:
```
'o'
```

In [55]:
```
tup4[-1]
```

Out[55]:
```
'three'
```

In [56]:
```
tup5[-1]
```

Out[56]:
```
(345, 98)
```

In [ ]:
```

```

In [ ]:
```

```

Tuple Slicing

In [57]:
```
mytuple = ('one' , 'two' , 'three' , 'four' , 'five' , 'six' , 'seven' , 'eight'
```

In [58]:
```
mytuple[:3]
```

Out[58]:
```
('one', 'two', 'three')
```

In [59]:
```
mytuple[2:5]
```

Out[59]: ('three', 'four', 'five')

In [60]:
```
mytuple[:8]
```

Out[60]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [61]:
```
mytuple[-3:]
```

Out[61]: ('six', 'seven', 'eight')

In [62]:
```
mytuple[:]
```

Out[62]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [ ]:

Remove & Change Items

In [63]:
```
mytuple
```

Out[63]: ('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')

In [65]:
```
del mytuple[0] # Tuples are immutable which means we can't DELETE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[65], line 1
----> 1 del mytuple[0] # Tuples are immutable which means we can't DELETE tuple i
tems

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

In [66]:
```
mytuple[0]=1  # Tuples are immutable which means we can't CHANGE tuple items
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[66], line 1
----> 1 mytuple[0]=1  # Tuples are immutable which means we can't CHANGE tuple it
ems

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

In [67]:
```
del mytuple  # Deleting entire tuple object is possible
```

In [68]:
```
mytuple
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[68], line 1
----> 1 mytuple

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

In [ ]:

Loop through a Tuple

In [69]:
```
mytuple =('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [70]:
```
mytuple
```

Out[70]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [71]:
```
for i in mytuple:
    print(i)
```

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

In [73]:
```
for i in enumerate(mytuple):
    print(i)
```

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

In [ ]:
```
```

Count

In [74]:
```
mytuple1 =('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
```

In [75]:
```
mytuple1.count('one')
```

Out[75]:
```
3
```

In [76]:
```
mytuple1.count('two')
```

Out[76]:
```
2
```

In [77]:
```
mytuple1.count('three')
```

Out[77]:
```
2
```

In [ ]:
```
```

Tuple membership

In [78]:
```
mytuple
```

Out[78]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [79]:
```
'one' in mytuple
```

Out[79]:
```
True
```

In [80]:
```
'Eleven' in mytuple
```

Out[80]:
```
False
```

In [81]:
```
if 'four' in mytuple:
    print('four is present in the tuple')
else:
    print('four is not present in the tuple')
```

```
four is present in the tuple
```

In [ ]:
```

```

Index position

In [83]:
```
mytuple
```

Out[83]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

In [84]:
```
mytuple.index('one')
```

Out[84]:
```
0
```

In [85]:
```
mytuple.index('six')
```

Out[85]:
```
5
```

In [ ]:
```

```

Sortinng

In [86]:
```
mytuple2=(45,23,7,27,12,98,34)
```

In [87]:
```
sorted(mytuple2)
```

Out[87]:
```
[7, 12, 23, 27, 34, 45, 98]
```

In [88]:
```
sorted(mytuple2,reverse=True)
```

Out[88]:
```
[98, 45, 34, 27, 23, 12, 7]
```

In [ ]:

In [ ]:

SET

1) Unordered & Unindexed collection of items.

2) Set elements are unique. Duplicate elements are not allowed.

3) Set elements are immutable (cannot be changed).

4) Set itself is mutable. We can add or remove items from it.

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

In [ ]:
```
s1={2,3,1,100,10}
s1
```

In [ ]:
```
s2=set()
type(s2)
```

In [22]:
```
s1={2,3,1,100,10}
s1
```

Out[22]: {1, 2, 3, 10, 100}

In [23]:
```
print(s1)
```

```
{1, 2, 3, 100, 10}
```

In [ ]:
```
print(type(s1))
```

In [ ]:
```
print(s)
print(type(s))
print(s1)
print(type(s1))
print(s2)
print(type(s2))
```

In [ ]:
```
s1
```

In [ ]:
```
s3={1,2.3,'nit',1+2j,True}
s3
```

In [ ]:
```
s1
```

In [ ]:
```
s1.add(4)
```

In [ ]:
```
s1
```

In [ ]:
```
s1.add('nit')
s1
```

In [ ]:
```
s1.add(10)
```

In [ ]:
```
s1
```

In [ ]:
```
s1.add([1,2,3])
```

In [ ]:
```
s4= s1.copy()
```

In [ ]:
```
s1==s4
```

In [ ]:
```
s1
```

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

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

In [ ]:
```
s1
```

In [ ]:
```
s1.pop()
```

In [ ]:
```
s1
```

In [ ]:
```
s1.pop()
s1
```

In [ ]:
```
s1.pop(2)
```

In [ ]:
```
s1
```

In [ ]:
```
s1.remove(100)
```

In [ ]:
```
s1
```

In [ ]:
```
s1.remove(200)
```

In [ ]:
```
s3
```

In [ ]:
```
s3.discard(200)
```

In [ ]:
```
s3.remove(200)
```

In [ ]:
```
s3
```

In [ ]:
```
s3.discard(1+2j)
```

In [ ]:
```
s3
```

In [89]:
```
s1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[89], line 1
----> 1 s1

NameError: name 's1' is not defined
```

In [91]:
```
myset ={1,3,8,4,3}
myset
```

Out[91]:
```
{1, 3, 4, 8}
```

In [92]:
```
len(myset)
```

Out[92]:
```
4
```

In [93]:
```
my_set = {1,1,2,2,3,4,5,5}
my_set
```

Out[93]:
```
{1, 2, 3, 4, 5}
```

In [94]:
```
myset1 = {1.79,2.08,3.99,4.56,5.45} # Set of float numbers
myset1
```

Out[94]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

In [96]:
```
myset2 = {'Asif' , 'John' , 'Tyrion'} # Set of Strings
myset2
```

Out[96]:
```
{'Asif', 'John', 'Tyrion'}
```

In [99]:
```
myset3 = {10,20, "Hola", (11, 22, 32)} # Mixed datatypes
myset3
```

Out[99]:
```
{(11, 22, 32), 10, 20, 'Hola'}
```

In [100…
```
myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items like li
myset3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[100], line 1
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]} # set doesn't allow mutable items
like li
      2 myset3

TypeError: unhashable type: 'list'
```

In [102…
```
myset4 = set() # Create an empty set Interview Question
print(type(myset4))
```

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

In [103…
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set
```

Out[103… {1, 2, 3, 4, 5}

In [105…
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
for i in myset:
    print(i)
```

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

In [106…
```
for i in enumerate(myset):
    print(i)
```

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

In [107…
```
myset
```

Out[107… {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [108…
```
'one' in myset
```

Out[108… True

In [109…
```
'ten' in myset
```

Out[109… False

In [ ]:

Add & Remove items

In [110…
```
myset
```

Out[110… {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [111…
```
myset.add('NINE')
myset
```

Out[111… {'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

In [112…
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE']) # Add multiple item to a set using
myset
```

Out[112…
```
{'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [113…
```
myset.remove('NINE')
myset
```

Out[113…
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [114…
```
myset.discard('TEN')
myset
```

Out[114…
```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

In [115…
```
myset.clear()
myset
```

Out[115…
```
set()
```

In [117…
```
del myset
myset
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[117], line 1
----> 1 del myset
      2 myset

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

In [ ]:

Copy

In [118…
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

Out[118…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [119…
```
myset1=myset
```

In [120…
```
myset1
```

Out[120…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [121…
```
id(myset)
```

Out[121…
```
2576748729440
```

In [122…
```
id(myset)
```

Out[122…
```
2576748729440
```

In [123…
```
my_set=myset.copy()
my_set
```

Out[123…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [124…
```
id(my_set)
```

Out[124…
```
2576745098240
```

In [125…
```
myset.add('nine')
myset
```

Out[125…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [126…
```
myset1
```

Out[126…
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

In [127…
```
my_set
```

Out[127…
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

set of operations

In [128…
```
s5 ={1,2,3,4,5}
s6={4,5,6,7,8}
s7={8,9,10}
```

In [ ]:
```

```

In [129…
```
s5.union(s6)
```

Out[129…
```
{1, 2, 3, 4, 5, 6, 7, 8}
```

In [130…
```
s5.union(s6,s7)
```

Out[130…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [131…
```
s5 | s6 | s7
```

Out[131…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

In [132…
```
s7|s6|s5
```

Out[132…
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

intersection

In [133…
```
s5 ={1,2,3,4,5}
s6={4,5,6,7,8}
s7={8,9,10}
```

In [134…
```
s5.intersection(s6)
```

Out[134…
```
{4, 5}
```

In [135…
```
s6.intersection(s7)
```

Out[135…
```
{8}
```

In [136…
```
s6 & s7
```

Out[136…
```
{8}
```

In [137…
```
s5 & s7
```

Out[137…
```
set()
```

Difference

In [10]:
```
s5 ={1,2,3,4,5}
s6={4,5,6,7,8}
s7={8,9,10}
```

In [11]:
```
s5.difference(s6)
```

Out[11]: {1, 2, 3}

In [12]:
```
s6.difference(s5)
```

Out[12]: {6, 7, 8}

In [13]:
```
s5 - s6
```

Out[13]: {1, 2, 3}

In [14]:
```
s6-s5
```

Out[14]: {6, 7, 8}

In [15]:
```
s5.difference(s7)
```

Out[15]: {1, 2, 3, 4, 5}

In [16]:
```
s6-s7
```

Out[16]: {4, 5, 6, 7}

In [140…
```
s5 ={1,2,3,4,5}
s6={4,5,6,7,8}
s7={8,9,10}
```

In [141…
```
s5.symmetric_difference(s6) ## Symmetric difference (Set of elements in A and B
```

Out[141… {1, 2, 3, 6, 7, 8}

In [142…
```
"""
Updates the set calling the symmetric_difference_update() method with the symmet
For below example Set A will be updated with the symmetric difference of s5 & s6
"""
s5.symmetric_difference_update(s6)
```

In [ ]:

In [143…
```
s5
```

Out[143… {1, 2, 3, 6, 7, 8}

In [21]:
```
print(s5)
print(s6)
print(s7)
```

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

Superset,subset ,disjoint

In [1]:
```
ss1={1,2,3,4,5,6,7,8,9}
ss2={3,4,5,6,7,8}
```

```
ss3={10,20,30,40}
```

In [2]:
```
ss1.issuperset(ss2)
```

Out[2]: True

In [3]:
```
ss2.issubset(ss1)
```

Out[3]: True

In [4]:
```
ss1.issubset(ss2)
```

Out[4]: False

In [5]:
```
print(ss1)
print(ss2)
print(ss3)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{3, 4, 5, 6, 7, 8}
{40, 10, 20, 30}
```

In [144…
```
ss3.isdisjoint(ss2)   # Two sets are said to be disjoint sets if they have no co
```

Out[144… True

In [8]:
```
ss3.isdisjoint(ss1)
```

Out[8]: True

In [9]:
```
ss3.isdisjoint(ss2)
```

Out[9]: True

In [10]:
```
ss4={1,2,3,4,5}
ss5={6,7,8}
ss6={9,10}
```

In [11]:
```
ss4.isdisjoint(ss5)
```

Out[11]: True

In [146…
```
ss4.issuperset(ss5)
```

Out[146… False

In [147…
```
sum(ss3)
```

Out[147… 100

In [148…
```
max(ss3)
```

Out[148… 40

In [149…
```
min(ss3)
```

Out[149… 10

In [150…
```
len(ss3)
```

Out[150… 4

In [151…
```
list(enumerate(ss3))
```

Out[151… [(0, 40), (1, 10), (2, 20), (3, 30)]

In [152…
```
sor=sorted(ss3,reverse=True)
sor
```

Out[152… [40, 30, 20, 10]

In [153…
```
sorted(sor)
```

Out[153… [10, 20, 30, 40]

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

set is completed

Dictionary

```
### Dictionary is a mutable data type in Python.
### A python dictionary is a collection of key and value pairs
separated by a colon (:) & enclosed in curly braces {}.
### Keys must be unique in a dictionary, duplicate values are
allowed.
```

In [13]:
```
d={}
type(d)
```

Out[13]: dict

In [14]:
```
dict()
```

Out[14]: {}

In [15]:
```
set()
```

Out[15]: set()

In [16]:
```
tuple()
```

Out[16]:
```
()
```

In [18]:
```
list()
```

Out[18]:
```
[]
```

In [164…
```
mydict ={1:'one',2:'two',3:'three'}
mydict
```

Out[164…
```
{1: 'one', 2: 'two', 3: 'three'}
```

In [21]:
```
mydict.keys()
```

Out[21]:
```
dict_keys([1, 2, 3])
```

In [22]:
```
mydict.values()
```

Out[22]:
```
dict_values(['one', 'two', 'three'])
```

In [23]:
```
mydict.items()
```

Out[23]:
```
dict_items([(1, 'one'), (2, 'two'), (3, 'three')])
```

In [24]:
```
mydict[4]='Four'
```

In [25]:
```
mydict
```

Out[25]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'Four'}
```

In [26]:
```
mydict['five']=5
mydict[2.3]='float'
mydict[1+2j]='complex'
```

In [27]:
```
mydict
```

Out[27]:
```
{1: 'one',
 2: 'two',
 3: 'three',
 4: 'Four',
 'five': 5,
 2.3: 'float',
 (1+2j): 'complex'}
```

In [166…
```
keys={'a','b','c','d'}
mydict3=dict.fromkeys(keys)
mydict3
```

Out[166…
```
{'d': None, 'b': None, 'a': None, 'c': None}
```

In [167…
```
print(type(keys))
print(type(mydict3))
```

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

In [169…
```
keys={'a','b','c','d'}
value =10
mydict3=dict.fromkeys(keys,value)
mydict3
```

Out[169…
```
{'d': 10, 'b': 10, 'a': 10, 'c': 10}
```

In [171…
```
keys={'a','b','c','d'}
value =[10,20,30]
mydict3=dict.fromkeys(keys,value)
mydict3
```

Out[171…
```
{'d': [10, 20, 30], 'b': [10, 20, 30], 'a': [10, 20, 30], 'c': [10, 20, 30]}
```

In [172…
```
value.append(40)
mydict3
```

Out[172…
```
{'d': [10, 20, 30, 40],
 'b': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
 'c': [10, 20, 30, 40]}
```

In [173…
```
mydict[1]
```

Out[173…
```
'one'
```

In [174…
```
mydict.get(1)
```

Out[174…
```
'one'
```

In [ ]:
```
```

Add, Remove & Change Items

In [176…
```
mydict1={'Name' :'Ravi', 'ID':56788,'DOB':1996,'Address':'India'}
mydict1
```

Out[176…
```
{'Name': 'Ravi', 'ID': 56788, 'DOB': 1996, 'Address': 'India'}
```

In [177…
```
mydict1['DOB']= 1997
mydict1['Address']='Bangalore'
mydict1
```

Out[177…
```
{'Name': 'Ravi', 'ID': 56788, 'DOB': 1997, 'Address': 'Bangalore'}
```

In [178…
```
dict1 ={'DOB':1995}
mydict1.update(dict1)
mydict1
```

Out[178…
```
{'Name': 'Ravi', 'ID': 56788, 'DOB': 1995, 'Address': 'Bangalore'}
```

In [179…
```
mydict1['Job']='Analyst'
mydict1
```

Out[179…
```
{'Name': 'Ravi',
 'ID': 56788,
 'DOB': 1995,
 'Address': 'Bangalore',
 'Job': 'Analyst'}
```

In [180…
```
mydict1.pop('Job')
mydict1
```

Out[180…
```
{'Name': 'Ravi', 'ID': 56788, 'DOB': 1995, 'Address': 'Bangalore'}
```

In [181…
```
mydict1.popitem()
```

Out[181…
```
('Address', 'Bangalore')
```

In [182…
```
del[mydict1['ID']]
mydict1
```

Out[182…
```
{'Name': 'Ravi', 'DOB': 1995}
```

In [183…
```
mydict1.clear()
mydict1
```

Out[183…
```
{}
```

In [184…
```
del mydict1
mydict1
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[184], line 2
      1 del mydict1
----> 2 mydict1

NameError: name 'mydict1' is not defined
```

In [ ]:

In [ ]:

Range

In [154…
```
range()
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[154], line 1
----> 1 range()

TypeError: range expected at least 1 argument, got 0
```

In [155…
```
range(5)
```

Out[155…
```
range(0, 5)
```

In [157…
```
list(range(0,5))
```
Out[157…
```
[0, 1, 2, 3, 4]
```
In [ ]:
```
range(10, 20)
```
In [158…
```
list(range(10, 20))
```
Out[158…
```
[10, 11, 12, 13, 14, 15, 16, 17, 18, 19]
```
In [159…
```
range(0, 100,20)  #Range (Start index,end index,step)
```
Out[159…
```
range(0, 100, 20)
```
In [160…
```
list(range(0, 100,10))
```
Out[160…
```
[0, 10, 20, 30, 40, 50, 60, 70, 80, 90]
```
In [161…
```
list(range(0, 100,20))
```
Out[161…
```
[0, 20, 40, 60, 80]
```
In [163…
```
range(0, 100,20,5) #range expected at most 3 arguments, got 4
```
```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[163], line 1
----> 1 range(0, 100,20,5) #range expected at most 3 arguments, got 4

TypeError: range expected at most 3 arguments, got 4
```
In [ ]:

Bit(Binary number system)

In [ ]:
```
'''
 binary (base 2)  -- 0& 1 (Ob)
 octal (base 8) -- 1,2,3,4,5,6,7(Oo)
 decimal (base 10)--Ox
 hexadecimal (base 16) -- Ox a-10,b-11,c-12,d-13,e-14,f-15 (0-9 a-f)
'''
```
In [ ]:

In [ ]:

In [ ]:

In [ ]: