Dictionary

```
month = {}
month[1] = 'Jan'
month[2] = 'Feb'
month[3] = 'Mar'
month[4] = 'Apr'
print(month)
type(month)
```

```
{1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr'}
dict
```

```
price = {'tomato':40, 'cucumber':30, 'potato':20, 'cauliflower':70, 'cabbage':50, 'lettuce':40, 'rsddish':30, '
a = price['potato']
b = price['carrot']
print(a)
print(b)
```

```
20
20
```

```
price = {'tomato':40, 'cucumber':30, 'potato':20, 'cauliflower':70, 'cabbage':50, 'lettuce':40, 'rsddish':30, '
price.keys()
```

```
dict_keys(['tomato', 'cucumber', 'potato', 'cauliflower', 'cabbage', 'lettuce', 'rsddish', 'carro
```

```
price = {'tomato':40, 'cucumber':30, 'potato':20, 'cauliflower':70, 'cabbage':50, 'lettuce':40, 'rsddish':30, '
price.values()
```

```
dict_values([40, 30, 20, 70, 50, 40, 30, 20, 80])
```

```
price = {'tomato':40, 'cucumber':30, 'potato':20, 'cauliflower':70, 'cabbage':50, 'lettuce':40, 'rsddish':30, '
price.items()
```

```
dict_items([('tomato', 40), ('cucumber', 30), ('potato', 20), ('cauliflower', 70), ('cabbage', 50
```

```
price = {'tomato':40, 'cucumber':30, 'potato':20, 'cauliflower':70, 'cabbage':50, 'lettuce':40, 'rsddish':30, '
price['tomato'] = 25
print(price)
```

```
{'tomato': 25, 'cucumber': 30, 'potato': 20, 'cauliflower': 70, 'cabbage': 50, 'lettuce': 40, 'rs
```

Dictionary Operators

```
digits = {0:'Zero', 1:'One', 2:'Two', 3:'Three', 4:'Four', 5:'Five', 6:'Six', 7:'Seven', 8:'Eight', 9:'Nine'}
a = len(digits) #Length operstor len (number of key-value pairs in a dictionary)
b = digits[1] #Indexing
c = min(digits) #Function min
d = max(digits) #Function max
e = sum(digits) #Function sum (assuming keys are compatible for addition)
f = 5 in digits #Membership operator in
g = 'Five' in digits #Membership operator in
print('a =',a)
print('b =',b)
print('c =',c)
print('d =',d)
print('e =',e)
print('f =',f)
print('g =',g)
```

```
a = 10
b = One
c = 0
d = 9
e = 45
f = True
g = False
```

```
winter = {11:'November', 12:'December', 1:'January', 2:'Febraury'}
2 in winter, min(winter), max(winter), sum(winter)
```

```
(True, 1, 12, 26)
```

```
winter = {11:'November', 12:'December', 1:'January', 2:'Febraury'}
2 in winter.keys(), min(winter.keys()), max(winter.keys()), sum(winter.keys())
```

```
(True, 1, 12, 26)
```

Deletion

```
winter = {11:'November', 12:'December', 1:'January', 2:'Febraury'}
del winter[11]
print(winter)
```

```
{12: 'December', 1: 'January', 2: 'Febraury'}
```

```
winter = {11:'November', 12:'December', 1:'January', 2:'Febraury'}
months = winter
months.clear()
months, winter
```

```
({}, {})
```

Function get

```
passwords = {'Ram':'ak@607','Shyam':'rou.589','Gita':'yam@694'}
passwords.get('Ram',-1)
```

```
passwords = {'Ram':'ak@607','Shyam':'rou.589','Gita':'yam@694'}
passwords.get('Raman',-1)
```

```
-1
```

```
passwords = {'Ram':'ak@607','Shyam':'rou.589','Gita':'yam@694'}
print(passwords.get('Raman'))
```

```
None
```

Function update

```
passwards = {'Ram':'ak@607','Shyam':'rou.589','Gita':'yam@694'}
morePasswords = {'Raman':'vi97@4','Kishore':'23@0jsk'}
passwords.update(morePasswords)
passwords
```

```
{'Gita': 'yam@694',
 'Kishore': '23@0jsk',
 'Ram': 'ak@607',
 'Raman': 'vi97@4',
 'Shyam': 'rou.589'}
```

Function copy

```
passwards = {'Ram':'ak@607','Shyam':'rou.589','Gita':'yam@694'}
morePasswords = {'Raman':'vi97@4','Kishore':'23@0jsk'}
passwords.update(morePasswords)
newPasswords = morePasswords.copy()
print(newPasswords)
print(morePasswords)
id(newPasswords), id(morePasswords)
```

```
{'Raman': 'vi97@4', 'Kishore': '23@0jsk'}
{'Raman': 'vi97@4', 'Kishore': '23@0jsk'}
(139758737183368, 139758737241720)
```

List of Functions

```
D.items()   #Return an object comprising of tuples of key-values pairs present in dictionary D.
D.keys()    #Return an object comprising of all keys of dictionary D.
D.values()  #Return an object comprising of all values of dictionary D.
D.clear()   #Return all key-value pairs from dictionary D.
D.get(key,default)  #For the specified key, the function returns the associated value. Returns the default valu
D.copy      #Creates a shallow copy of dictionary D.
D1.updates(D2)  #Adds the key-value pairs of dictionary D2 to dictionary D1.
```

Inverted Dictionary

```
def buildInvDict(dic1):
  '''
  objective: To construct inverted dictionary
  Input Parameter: dict1 : dictionary
  Return Value: invDict : dictionary
  '''
  invDict = ()
  for key,value in dict1.items():
    if value in invDict:
      invDict[value].append(key)
    else:
      invDict[value]=[key]
  invDict = {x:invDict[x] for x in invDict if len(invDict[x])>1}
  return invDict

def main():
  '''
  objective: To find inverted dictionary
  Input Parameter: None
  Return Value: None
  '''
  wordMeaning = eval(input('Enter word meaning dictionary: '))
  meaningWord = buildInvDict(wordMeaning)
  print('Inverted Dictionary:\n',meaningWord)

#Statements to initiate the call to main function
if __name__ == '__main__':
    main()
```

```
#Program to take names and salary of the employees

num = int(input("Enter the number of employees to be stored: "))
count = 1
employee = dict()   #create an empty dictionary
while count <= num:
  name = input("Enter the name of the Employees: ")
  salary = int(input("Enter the salary: "))
  employee[name] = salary
  count += 1
print("\n\nEMPLOYEE_NAME\t\tSALARY")
for k in employee:
  print(k,'\t\t',employee[k])
```

```
Enter the number of employees to be stored: 3
Enter the name of the Employees: Ankush Rana
Enter the salary: 400000
Enter the name of the Employees: Ujjwal Jaryal
Enter the salary: 200000
Enter the name of the Employees: Surbhi Jarwal
Enter the salary: 100000


EMPLOYEE_NAME           SALARY
Ankush Rana              400000
Ujjwal Jaryal            200000
Surbhi Jarwal            100000
```