In [ ]:
```
#CREATE A DICTIONARY-1) curly braces, {}   2) built-in function  dict
() function.
#1)CURLY BRACES
mydict={}
print (type(mydict))

#2)using dict()
mydict=dict()
print (type(mydict))
```

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

In [ ]:
```
# keys in the dictionary are Boolean, integer, floating point number,
 and string data types, which are all acceptable.Dictionary keys canno
t be of a type that is mutable, such as sets, lists, or dictionaries.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}

print(my_dictionary)
#key with list type are not supported
my_dictionary = {["Python"]: "languages"}

print(my_dictionary)
```

```
{True: 1, 1.1: 1.1, 'one': 1, 'languages': ['Python']}

---------------------------------------------------------------------
----
TypeError                                 Traceback (most recent call l
ast)
<ipython-input-1-7ed421803986> in <module>()
      4 print(my_dictionary)
      5 #key with list type are not supported
----> 6 my_dictionary = {["Python"]: "languages"}
      7
      8 print(my_dictionary)

TypeError: unhashable type: 'list'
```

In [ ]:
```
#creating a dictionary with items using {}
mydict={'name':"john",'age':45,'job':"doctor"}
print(mydict)
#create a dictionary with dict()
mydict = dict({'name': 'john' ,'age':45,'job':"doctor"})
print(mydict)
```

```
{'name': 'john', 'age': 45, 'job': 'doctor'}
{'name': 'john', 'age': 45, 'job': 'doctor'}
```

In [ ]:
```
#creating a dictionary using fromkeys() without setting a value for al
l the keys:
#create sequence of strings
cities = ('Paris','Athens', 'Madrid')
#create the dictionary, `my_dictionary`, using the fromkeys() method
my_dictionary = dict.fromkeys(cities)
print(my_dictionary)
```

```
{'Paris': None, 'Athens': None, 'Madrid': None}
```

In [ ]:
```
#create a sequence of strings
cities = ('Paris','Athens', 'Madrid')
#create a single value
continent = ('Europe')
my_dictionary = dict.fromkeys(cities,continent)

print(my_dictionary)
```

```
{'Paris': 'Europe', 'Athens': 'Europe', 'Madrid': 'Europe'}
```

In [ ]:
```
#len() function returns the total length of the object that is passed
 as an argument.(no of key value pair)
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (len(my_dictionary))
```

```
4
```

In [ ]:
```
#dictionaries can be created from list
L1=[1,2,3]
L2=[10,20,30]
dictionary1=dict(zip(L1,L2))
print (dictionary1)
```

```
{1: 10, 2: 20, 3: 30}
```

In [ ]:
```
#1)View All key-value Pairs Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.items())
#2)View All keys Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.keys())
#3)#View All values Contained in a Dictionary in Python
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.values())
```

```
dict_items([(True, 1), (1.1, 1.1), ('one', 1), ('languages', ['Pytho
n'])])
dict_keys([True, 1.1, 'one', 'languages'])
dict_values([1, 1.1, 1, ['Python']])
```

In [ ]:
```
#how to access an item in a Python dictionary:
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary["one"])
#key not in dictionary
print (my_dictionary["two"])
```

```
1

---------------------------------------------------------------------------
----
KeyError                                  Traceback (most recent call l
ast)
<ipython-input-24-0c92bab679da> in <module>()
      3 print (my_dictionary["one"])
      4 #key not in dictionary
----> 5 print (my_dictionary["two"])

KeyError: 'two'
```

In [ ]:
```
#in keyword  returns True if the key is in the dictionary and False if
it isn't.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print ("one" in my_dictionary)
print ("two" in my_dictionary)
```

```
True
False
```

In [ ]:
```
#Another way around this is to access items in the dictionary by using
the get() method.
my_dictionary = {True: "True",  1: 1,  1.1: 1.1, "one": 1, "languages"
: ["Python"]}
print (my_dictionary.get("one"))
print (my_dictionary.get("two","no such key"))
```

```
1
no such key
```

In [ ]:
```
#Add New Items to A Dictionary in Python
my_dictionary = {}

#add a key-value pair to the empty dictionary
my_dictionary['name'] = "John "
# add another  key-value pair
my_dictionary['age'] = 34

#print dictionary
print(my_dictionary)
my_dictionary['age'] = 46

#the value of 'age' will now be updated

print(my_dictionary)
```

```
{'name': 'John ', 'age': 34}
{'name': 'John ', 'age': 46}
```

In [ ]:
```
#To update a dictionary, you can also use the dictionary method update
() .
my_dictionary={'name': 'John ', 'age': 34}
my_dictionary.update(name= 'Mike Green', age = 46, occupation = "softw
are developer")
print(my_dictionary)
#update method to combine two dictionaries
numbers = {'one': 1, 'two': 2, 'three': 3}
more_numbers = {'four': 4, 'five': 5, 'six': 6}

#update 'numbers' dictionary
#you update it by adding the contents of another dictionary, 'more_num
bers',
#to the end of it
numbers.update(more_numbers)

print(numbers)
```

```
{'name': 'Mike Green', 'age': 46, 'occupation': 'software developer'}
{'one': 1, 'two': 2, 'three': 3, 'four': 4, 'five': 5, 'six': 6}
```

In [ ]:
```
#copy method
my_dictionary={'name': 'John ', 'age': 34}
newdict=my_dictionary.copy()
print(newdict)
```

```
{'name': 'John ', 'age': 34}
```

In [ ]:
```
#delete a key-value pair
my_dictionary={'name': 'John ', 'age': 34}
del my_dictionary['age']

print(my_dictionary)
#remove a key value pair and store it using pop method
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.pop('age')
print(value)
#pop with a default value
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.pop('job',"not in dictionary")
print(value)
#remove last item from dictionary-popitem
my_dictionary={'name': 'John ', 'age': 34}
value=my_dictionary.popitem()
#delete an element from dictionary
my_dictionary={'name': 'John ', 'age': 34}
del(my_dictionary['name'])
print(my_dictionary)
print(value)
```

```
{'name': 'John '}
34
not in dictionary
('age', 34)
```

In [ ]:
```
#delete all items-clear method
my_dictionary={'name': 'John ', 'age': 34}
my_dictionary.clear()
print(my_dictionary)
```

```
{}
```

In [ ]:
```
#1)traversing a dictionary
my_dictionary={'name': 'John ', 'age': 34}
print(my_dictionary)
for i in my_dictionary:
  print(i,my_dictionary[i])
#using items
my_dictionary={'name': 'John ', 'age': 34}
print(my_dictionary.items())
for i,j in my_dictionary.items():
  print (i,j)
```

```
{'name': 'John ', 'age': 34}
name John
age 34
dict_items([('name', 'John '), ('age', 34)])
name John
age 34
```

In [ ]:
```
#sort-convert the dictionary to a list and use sort method
my_dictionary={'name': 'John ', 'age': 34}
l1=list(my_dictionary.keys())
l1.sort()
print (l1)
for i in l1:
  print(i,my_dictionary[i])

#2)using build in function called sorted return value is sorted list o
f keys
my_dictionary={'name': 'John ', 'age': 34}
newdict=sorted(my_dictionary)
print(newdict)
```

```
['age', 'name']
age 34
name John
['age', 'name']
```

In [ ]:
```
#dictionary membership function
my_dictionary={'name': 'John ', 'age': 34}
print('name'in my_dictionary)
print('job'in my_dictionary)
```

```
True
False
```

In [ ]:
```
#Dictionary Comprehension
 squares = {x: x*x for x in range(6)}
 print(squares)
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

In [ ]:
```
#entering key-value pair at runtime
dict1={}
n=int(input("enter number of elements in dictionary"))
for i in range(0,n):
  key=int(input("key"))
  dict1[key]=int(input("value"))
print(dict1)
```

```
enter number of elements in dictionary5
key1
value2
key1
value5
key2
value2
key4
value2
key6
value1
{1: 5, 2: 2, 4: 2, 6: 1}
```

In [ ]:
```
#key-value at runtime
str1=input("enter the string")
dict1={}
for i in str1:
  dict1[i]=(int(input("enter value")))
print (dict1)
```

```
enter the stringhello
enter value1
enter value3
enter value1
enter value5
enter value7
{'h': 1, 'e': 3, 'l': 5, 'o': 7}
```

In [ ]:
```
#Program to count the number of occurrence(frequency) of each letters
 in a given string( histogram)
str1=input("enter the string")
dict1={}
for i in str1:
  dict1[i]=(dict1.get(i,0)+1)
print (dict1)
```

```
enter the stringhello
{'h': 1, 'e': 1, 'l': 2, 'o': 1}
```

In [ ]:
```
#Program to display the frequency of each word in a given string
str1=input("enter the string")
dictnw={}
l1=str1.split()
for i in l1:
  dictnw[i]= dictnw.get(i,0)+1
print(dictnw)
```

```
enter the stringhello hai hello
{'hello': 2, 'hai': 1}
```

In [ ]:
```
#Write a Python program to create a dictionary of roll numbers and nam
es of five students.
#sort the roll nubers(sorting by converting to a list)
dict1={}
for i in range(0,3):
  roll=int(input("roll"))
  dict1[roll]=input("name")
print (dict1)
l2=list(dict1)
print(l2)
l2.sort()
for i in l2:
  print(i,dict1[i])
```

```
roll2
namefg
roll1
namesd
roll11
namef
{2: 'fg', 1: 'sd', 11: 'f'}
[2, 1, 11]
1 sd
2 fg
11 f
```

In [ ]:
```
#sortig a dictionary using buildin sorted method
dict1={}
for i in range(0,3):
  roll=input("roll")
  dict1[roll]=input("name")
print (dict1)
for i in sorted(dict1):
  print(i,dict1[i])
```

```
roll4
namedf
roll1
namehj
roll4
namedd
{'4': 'dd', '1': 'hj'}
1 hj
4 dd
```

In [ ]:

```
#hex to binary conversion
hextobin={'0':'0000','1':'0001','2':'0010','3':'0011','4':'0100','5':
'0101','6':'0110','7':'0111','8':'1000','9':'1001','A':'1010','B':'101
1','C':'1100','D':'1101','E':'1110','F':'1111'}
n=input('Enter the hexadecimal number….')
bn=''
n=n.upper()
for i in n:
    h=hextobin.get(i)
    if h==None:
       print('Invalid Number')
       break
    print("binary equilavent of the heaxdecimal is {} is {}".format(n,
h))
```

```
Enter the hexadecimal number….A
binary equilavent of the heaxdecimal is A is 1010
```