```
#! /usr/bin/python3

times = 100

print ("[1]:Creating empty dictionary")
D = { }
print ("Empty dictionary D:", D)
print ("-" * times)

print ("[2]:Creating non-empty dictionary, Method 1")
D = {'Hello':1, 'Python':2, 'World':3}
print ("Non-empty dictionary D:", D)
print ("-" * times)

print ("[3]:Printing number of entries in dictionary")
print ("D:", D, " | len (D):", len (D))
print ("-" * times)

print ("[4]:Printing all keys in the dictionary")
print ("D.keys ():", D.keys ())
print ("Converting D.keys () output in list form")
D_key_list = list (D.keys ())
print ("D_key_list:", D_key_list, " | type (D_key_list):", type (D_key_list))
print ("-" * times)

print ("[5]:Printing all values in the dictionary")
print ("D.values ():", D.values ())
print ("Converting D.values () output in list form")
D_value_list = list (D.values ())
print ("D_value_list:", D_value_list, " | type (D_value_list):", type (D_value_list))
print ("-" * times)

print ("[6]:Accessing values by their keys")
print ("D['Hello']:", D['Hello'])
print ("D['Python']:", D['Python'])
print ("D['World']:", D['World'])
print ("-" * times)

print ("[7]:Printing values by iterating through dictionary")
for key in D.keys ():
    print ("D[", key, "]:", D[key])
print ("-" * times)

print ("[8]:Changing dictionary in-place")
print ("Original dictionary D:", D)
D['Python'] = 'PL'
print ("After D['Python'] = 'PL', D:", D)
D['Hello'] = [1,2,3]
print ('After D["Hello"] = [1,2,3], D:', D)
print ("-" * times)

print ("[9]:Deleting entries from the dictionary")
```

```
print ("Original dictionary D:", D)
del D['Hello']
print ("After del D['Hello'], D:", D)
del D['Python']
print ("After del D['Python'], D:", D)
del D['World']
print ("After del D['World'], D:", D)

print ("[10]:Using get method to get value from key")
D = {'Hello':1, 'Python':1, 'World':2}
print ("D.get('Hello'):", D.get ('Hello'))
print ("get method returns None if key does not exist")
if D.get ('FOO') == None:
    print ("Key 'FOO' does not exist in dictionary D:", D)
L_keys = ['Hello', 'Foo', 'Python', 'Bar', 'World', 'LOL']
for key in L_keys:
    if D.get (key) != None:
        print ("D[", key, "]:", D[key])
    else:
        print ("D[", key, "]:", key, "does not exist in list of keys of D", list
          (D.keys ()))
print ("-" * times)

print ("[11]:Update method to merge to dictionaries")
D1 = {"Hello":1, "Python":2, "World":3}
D2 = {"Perl":10, "Pro":20, "Lang":3}
print ("Original D1:", D1)
print ("Original D2:", D2)
D1.update (D2)
print ("After D1.update (D2), D1:", D1)
D1 = {'A':1, 'B':2, 'C':3, 'D':4}
D2 = {'A':10, 'B':20, 'E':5, 'F':6}
print ("Original D1:", D1)
print ("Original D2:", D2)
D1.update (D2)
print ("After D1.update (D2), D1:", D1)
D1 = {'A':1, 'B':2, 'C':3, 'D':4}
D2 = {'A':10, 'B':20, 'E':5, 'F':6}
print ("After restoration, D1:", D1)
print ("After restoration, D2:", D2)
D2.update (D1)
print ("After D2.update (D1), D2:", D2)
print ("-" * times)

print ("[12]:Fetching an entire items and making a list of it")
D = {'Hello':1, 'Python':2, 'World':3}
print ("Original D:", D)
print (list (D.items ()))
print ("-" * times)

print ("[13]:Deleting an entry using pop method")
print ("Original D:", D)
```

```
D.pop ('Hello')
print ("After D.pop ('Hello'), D:", D)
D.pop ('World')
print ("After D.pop ('World'), D:", D)
D.pop ('Python')
print ("After D.pop ('Python'), D:", D)
print ("-" * times)

print ("[14]:Using dictionary as a list")
D = {}
# Not all members of 100 element list require preallocation.
# We can just assign values to indices are valid.
D[99] = "Hello"
print ("D:", D)
# Same can be done for multi-dimensional array.
M = {}
M [(1,1,1)] = "X=1, Y=1, Z=1"
M [(99, 322,123)] = "X=99 Y=322 Z=1"
print ("M[(1,1,1)]:", M[(1,1,1)])
print ("M[(99, 322, 123):", M[(99, 322, 123)])
print ("-" * times)

print ("[15]:Dictionary Creation Method 2")
D = {}
print ("Initial empty D:", D)
D['Hello'] = 1
print ("After D['Hello']=1, D:", D)
D['Python'] = 2
print ("After D['Python']=2, D:", D)
D['World'] = 3
print ("After D['World']=3, D:", D)
print ("-" * times)

print ("[16]:Dictionary creation Method 3")
D = {}
print ("Initial empty D:", D)
D = dict (name='xyz', age=22, marks=40.40)
print ("After D=dict (name='xyz', age=22, marks=40.40), D:", D)
print ("Creating list of dictionary records out of lists")
L1 = ["xyz", "pqr", "lmn"]
L2 = [20, 22, 24]
L3 = [45.65, 60.43, 80.51]
L  = []
print ("L1:", L1)
print ("L2:", L2)
print ("L3:", L3)
print ("Creating list L of dictionaries created from L1, L2, and L3")
for i in range (len (L1)):
    L.append (dict (name=L1[i], age=L2[i], marks=L3[i]))
print ("Printing entries in L:")
for d in L:
    print (d['name'], d['age'], d['marks'])
```

```
print ("-" * times)

print ("[17]:Dictionary creation Method 3")
print ("Creating dictionary from list of tuples")
L = [('a',1), ('b',2), ('c',3)]
print ("L:", L)
D = dict (L)
print ("D after D = dict (L), D:", D)
print ("-" * times)

print ("[18]:Dictionary Creation Method 4")
L1 = ['a', 'b', 'c']
L2 = [1, 2, 3]
print ("L1:", L1, "L2:", L2)
L  = list (zip (L1, L2))
print ("L after L = list (zip (L1, L2)), L:", L)
D = dict (L)
print ("D after D = dict (L), D:", D)
print ("In single statement we can write:")
print ("D = dict (list (zip (L1, L2)))")
D = dict (list (zip (['Hello', 'Python', 'World'], [1,2,3])))
print ("D:", D)
print ("Greater flexibility can be obtained by using")
print ("list comprehensions while specifying L1 and L2")
print ("-" * times)

print ("[19]:Dictionary Creation Method 5")
print ("Create dictionary using dict.fromkeys")
print ("Useful when all keys are known and all values")
print ("have same initial value like 0 or None or any other")
D = dict.fromkeys (['S', 'P', 'A', 'M'], 0)
for key in D.keys ():
    print ("D[", key, "]:", D[key])
D = dict.fromkeys ([c for c in 'SPAM'], None)
for key in D.keys ():
    print ("D[", key, "]:", D[key])
D = dict.fromkeys ([c for c in 'SPAM'], 'v')
for key in D.keys ():
    print ("D[", key, "]:", D[key])
print ("-" * times)


print ("[20]:Dictionary comprehensions v1:Dictionary Creation Method 6")
print ("General form:D = {k:v for (k,v) in list (zip (L1, L2))}")
D = {k:v for (k,v) in list (zip ([c for c in 'SPAM'], [n for n in range (1, len
  ('SPAM')+1)]))}
print ("After executing:")
print ("D = {k:v for (k,v) in list (zip ([c for c in 'SPAM'], [n for n in range (1,
  len('SPAM')+1)]))}")
print ("D:", D)
D = {k:None for k in range (5)}
print ("After executing:")
```

```
print ("D={k:None for k in range (5)}")
print ("D:", D)
print ("-" * times)

print ("[21]:Dictionary comprehensions v2:Dictionary Creatio Method 7")
print ("General form: D = {x:f(x) for x in L}")
print ("After executing:")
print ("D = {n:n**2 for n in range (5)}")
D = {n:n**2 for n in range (5)}
print ("D:", D)
print ("-" * times)

print ("[22]:Accessing keys of dictionary in a sorted order")
print ("Constraints : list of keys in dictionary should be sortable")
D = {1:10, 5:20, 2:54, 3:987, 10:432}
print ("Accessing dictionary without sorting keys")
for key in D.keys ():
    print ("D[", key, "]:", D[key])
print ("Accessing dictionary with keys sorted")
for key in sorted (D.keys ()):
    print ("D[", key, "]:", D[key])
print ("-" * times)
```