tuple-set-dict-home-task

August 13, 2024

0.1 TUPLE

- declar with ()
- ex- variable_name = (val1, val2,val3,….)
- tuple is immutable
- inbuilt function (count,index)
- del and sorted function use to delete and sort the tuple element
- tuple not growable

0.1.1 Tuple creation

[264]:
```
t = () #empty tuple
type(t)
```

[264]:
```
tuple
```

[265]:
```
t = (1,2,3,4,5,6,7,8,9,0) # tuple with integer value
t
```

[265]:
```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
```

[266]:
```
t1 = (12.3,2.3,34.5,4.5,6.9) #tuple with float value
t1
```

[266]:
```
(12.3, 2.3, 34.5, 4.5, 6.9)
```

[267]:
```
t2 = ('sunil', 'ram', 'Hari', 'Krushna') # Tuple with string value
t2
```

[267]:
```
('sunil', 'ram', 'Hari', 'Krushna')
```

[268]:
```
t3 = (12+2j,34+4j,67+4j,2+3j) # tuple with complex value
t3
```

[268]:
```
((12+2j), (34+4j), (67+4j), (2+3j))
```

[269]:
```
t4 = (12,23.4,"darshani",12+3j,100) # tuple with mixed value
t4
```

[269]: (12, 23.4, 'darshani', (12+3j), 100)

[270]:
```
t5 = (12,34,(5.6,6,5,'Ram')) # tuple with nested tuple
t5
```

[270]: (12, 34, (5.6, 6, 5, 'Ram'))

[271]:
```
len(t5) # find lenth of the tuple
```

[271]: 3

0.1.2 Tuple indexing

[272]:
```
print(t)
```

(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)

[273]:
```
t[3] # retrive any index from the tuple
```

[273]: 4

[274]:
```
t5
```

[274]: (12, 34, (5.6, 6, 5, 'Ram'))

[275]:
```
t5[2][3] # nested  tuple indexing
```

[275]: 'Ram'

[276]:
```
t4
```

[276]: (12, 23.4, 'darshani', (12+3j), 100)

[277]:
```
t4[-3] # Negative indexing in tuple
```

[277]: 'darshani'

0.1.3 Tuple slicing

[278]:
```
t
```

[278]: (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)

[279]:
```
t[:]
```

[279]: (1, 2, 3, 4, 5, 6, 7, 8, 9, 0)

[280]:
```
t[2:6] # starting index is 2 last index in (6-1)
```

[280]:
```
(3, 4, 5, 6)
```

[281]:
```
t[-6:-1] # negative slicing
```

[281]:
```
(5, 6, 7, 8, 9)
```

Remove Change in Tuple

[282]:
```
t4
```

[282]:
```
(12, 23.4, 'darshani', (12+3j), 100)
```

[283]:
```
### As Tuple is immutable in Nature we can chnage any on the tuple
```

[284]:
```
t6 = [4,5,5,7,7]
```

[285]:
```
del t6 # Only we can delete the tuple if we want
```

[286]:
```
t6
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[286], line 1
----> 1 t6

NameError: name 't6' is not defined
```

0.1.4 Loop in tuple

[287]:
```
t
```

[287]:
```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 0)
```

[288]:
```
for i in t: # it iterate all the element from the tuple one by one
    print(i)
```

```
1
2
3
4
5
6
7
8
```

```
9
0
```

[289]:
```
for i in enumerate(t): #  The enumurate function give the pair value with the tuple
    print(i)
```

```
(0, 1)
(1, 2)
(2, 3)
(3, 4)
(4, 5)
(5, 6)
(6, 7)
(7, 8)
(8, 9)
(9, 0)
```

0.1.5 Tuple inbuilt function

- count
- index

[290]:
```
t6 = (12,21,12,34,56,56,12,9,"sunil", 56)
t6
```

[290]:
```
(12, 21, 12, 34, 56, 56, 12, 9, 'sunil', 56)
```

[291]:
```
t6.count(12) # count how muct time the element in the tuple occur
```

[291]:
```
3
```

[292]:
```
t6.index('sunil') # it gives the index according to the value or parameter
```

[292]:
```
8
```

0.1.6 tuple mumbership

- it use to check wheather the value in the tuple or not
- use in keyword
- ex- value in tuple_name

[293]:
```
t2
```

[293]:
```
('sunil', 'ram', 'Hari', 'Krushna')
```

[294]:
```
"Hari" in t2 # if the value inside the tuple then its return true other wise return false
```

[294]: True

[295]:
```
3 in t2
```

[295]: False

0.1.7 Sorting in Tuple

[296]:
```
t7 = (45,65,78,3,45,2,34,67,99,9)
```

[297]:
```
sorted(t7) # sorted function sort the element of tuple in ascending order
```

[297]: [2, 3, 9, 34, 45, 45, 65, 67, 78, 99]

[298]:
```
sorted(t7,reverse=True) # use this syntax to sort the tuple in descending order
```

[298]: [99, 78, 67, 65, 45, 45, 34, 9, 3, 2]

1 SET

- Set declar by {}
- to declar empty set use set_name = set()
- Duplicate are not allowed in set

[299]:
```
d = {}
type(d)
```

[299]: dict

[300]:
```
s = set() # creating empty set
type(s)
```

[300]: set

[301]:
```
s1 = {1,2,3,4,5,6,7,8,9} # set with integer
s2 = {1.2,2.3,4.5,5.6} # set with float
s3 = {"s",'df','sunil','hu'} #set with string
s4 = {10+2j,3+4,3+45j,5+9j} # set with complex number
s5 = {12,3.4,"du",10+3j} # set with mixed value
```

[302]:
```
print(s1)
print(s2)
print(s3)
print(s4)
print(s5)
```

```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
{1.2, 2.3, 4.5, 5.6}
{'s', 'hu', 'sunil', 'df'}
{(10+2j), (3+45j), (5+9j), 7}
{3.4, 12, (10+3j), 'du'}
```

[303]:
```
len(s1) # length of the set
```

[303]:
```
9
```

[304]:
```
s6 = {1,22,3,4,3,3,3,3,4,5,5,5,6} # duplicate are not allowed
s6
```

[304]:
```
{1, 3, 4, 5, 6, 22}
```

[305]:
```
s3
```

[305]:
```
{'df', 'hu', 's', 'sunil'}
```

[306]:
```
s8 = {1,2,[4,5,6]} # canot add list inside a set its unhashable or immutable
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[306], line 1
----> 1 s8 = {1,2,[4,5,6]} # canot add list inside a set its unhashable or␣
 ↪immutable

TypeError: unhashable type: 'list'
```

[307]:
```
t3
```

[307]:
```
((12+2j), (34+4j), (67+4j), (2+3j))
```

1.0.1 for loop in set

[308]:
```
for i in t3:
    print(i)
```

```
(12+2j)
(34+4j)
(67+4j)
(2+3j)
```

[309]:
```
for i in enumerate(t3):
    print(i)
```

```
(0, (12+2j))
(1, (34+4j))
(2, (67+4j))
(3, (2+3j))
```

1.0.2 set membership

[310]:
```
s2
```

[310]:
```
{1.2, 2.3, 4.5, 5.6}
```

[311]:
```
5.6 in s2
```

[311]:
```
True
```

1.0.3 membership using if else statement

[312]:
```
if 4.5 in s2:
    print("yes")
else:
    print('no')
```

```
yes
```

[313]:
```
if 4 in s2:
    print("yes")
else:
    print('no')
```

```
no
```

1.0.4 inbult function in set

- add(),remove(), clear(), copy()

[314]:
```
s1
```

[314]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

[315]:
```
s1.add(10) # add an element on the set
s1
```

[315]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
```

[316]:
```
s1.remove(10) #remove the element thet you give as argument
s1
```

[316]:
```
{1, 2, 3, 4, 5, 6, 7, 8, 9}
```

[317]:
```
s1.clear() # clear all the elment of the set
```

[318]:
```
s1
```

[318]:
```
set()
```

[319]:
```
s1.add(1,2,3,4,5,6,7,8,9) # we canot pass more the one argument in add function
s1
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[319], line 1
----> 1 s1.add(1,2,3,4,5,6,7,8,9) # we canot pass more the one argument in add␣
 ↪function
      2 s1

TypeError: set.add() takes exactly one argument (9 given)
```

[321]:
```
s1.add(1) # to add multiple element on set
s1.add(2)
s1.add(3)
s1.add(4)
s1.add(5)
s1.add(6)
s1
```

[321]:
```
{1, 2, 3, 4, 5, 6}
```

[322]:
```
s9 =s1.copy() #copy all the element of s1 to s9
s9
```

[322]:
```
{1, 2, 3, 4, 5, 6}
```

[323]:
```
s1
```

[323]:
```
{1, 2, 3, 4, 5, 6}
```

1.1 Set Operation

- union –> it combine all the element into one set taking duplicate at once
- intersection –>it show up only the common element thats are present on the two or more than two set
- difference –> it show only the alement present in a set excepting the common element
- symmetric_difference –> it show all the element of the sets excepting the common element
- issubset –> if a set all element is in the b set then a is subset of b
- issuperset –> if some of the eelment of a are in b then a superset b
- isdisjoint –> if no one common element in the both set it return true other wise false

[324]:
```
set1 = {1,2,3,4,5,6}
set2 = {5,6,7,8,9}
```

[325]:
```
print(set1)
print(set2)
```

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

[326]:
```
set1.union(set2) # combine all the element taking duplicate at once
```

[326]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[327]:
```
set1 | set2
```

[327]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[328]:
```
set1.intersection(set2) # take only the common element
```

[328]: {5, 6}

[329]:
```
set1 & set2
```

[329]: {5, 6}

[330]:
```
set1.difference(set2) # it show only the alement present in  set1 excepting the␣
↪common element
```

[330]: {1, 2, 3, 4}

[331]:
```
set1 - set2
```

[331]: {1, 2, 3, 4}

[332]:
```
set2 - set1
```

[332]: {7, 8, 9}

[333]:
```
set1.symmetric_difference(set2) # except the common element all element
```

[333]: {1, 2, 3, 4, 7, 8, 9}

[334]:
```
print(set1)
print(set2)
```

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

[335]:
```
set2.add(1)
set2.add(2)
set2.add(3)
set2.add(4)
set2.add(5)
set2
```

[335]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[336]:
```
print(set1)
print(set2)
```

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

[337]:
```
set1.issubset(set2) # all set1 element are in set2
```

[337]: True

[338]:
```
set2.issuperset(set1) # is in set2 some element of set1 are there
```

[338]: True

[339]:
```
set1
```

[339]: {1, 2, 3, 4, 5, 6}

[163]:
```
set2
```

[163]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[340]:
```
set2.remove(1) # remove the value from set
set2.remove(2)
set2.remove(3)
set2.remove(4)
set2.remove(5)
set2.remove(6)
set2
```

[340]: {7, 8, 9}

[341]:
```
set1.isdisjoint(set2) # set1 element is different in set2 element
```

[341]: True

1.1.1 other builtin fumction

-sum(), max(), min(), len(), sorted(),

[346]:
```
s1
```

[346]:
```
{1, 2, 3, 4, 5, 6}
```

[347]:
```
sum(s1) # find the sum of the set value
```

[347]:
```
21
```

[348]:
```
max(s1) # find the maximum value on set
```

[348]:
```
6
```

[349]:
```
min(s1) # find mminimun value on set
```

[349]:
```
1
```

[350]:
```
sorted(s1) # sort set value in ascending order
```

[350]:
```
[1, 2, 3, 4, 5, 6]
```

[351]:
```
sorted(s1,reverse=True) # sort set value in descending order
```

[351]:
```
[6, 5, 4, 3, 2, 1]
```

1.2 Dictionary

- declar with {}
- asign value in dictionary with key and value pair format
- ex - dic_name = {key1 : val1, key2: val2,....}

[185]:
```
d={} # creating empty dictionary
type(d)
```

[185]:
```
dict
```

[353]:
```
dic = {1:"One", 2: 'Two', 3:"Three",4:"Four",5:"Five",6:"Six"}
```

[354]:
```
dic
```

[354]:
```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five', 6: 'Six'}
```

[355]:
```
type(dic)
```

[355]:
```
dict
```

[356]:
```
dic.keys() # gives the keys of dictionary
```

[356]:
```
dict_keys([1, 2, 3, 4, 5, 6])
```

[357]:
```
dic.values()  # gives the values of the dictionary
```

[357]:
```
dict_values(['One', 'Two', 'Three', 'Four', 'Five', 'Six'])
```

[358]:
```
dic.items() # gives key and valus combination of the dictionary
```

[358]:
```
dict_items([(1, 'One'), (2, 'Two'), (3, 'Three'), (4, 'Four'), (5, 'Five'), (6,
'Six')])
```

[359]:
```
for i  in dic:
    print(dic[i])
```

```
One
Two
Three
Four
Five
Six
```

[360]:
```
dict=dic.copy() # copy the items from one dicstionary to another
```

[361]:
```
dict
```

[361]:
```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five', 6: 'Six'}
```

[362]:
```
dic.clear() # clear all the items from the dictionary
```

[205]:
```
dic
```

[205]:
```
{}
```

[363]:
```
dict.get(3) # give the corresponding values of the keys
```

[363]:
```
'Three'
```

[364]:
```
dict.pop(6) # pop out corresponding values of the key
```

[364]:
```
'Six'
```

[365]:
```
dict
```

[365]:
```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five'}
```

[366]:
```
d = {1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five'}
```

[367]:
```
d
```

[367]:
```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five'}
```

[368]:
```
keys = 1,2,3,4,5,6,7,8,9 #according keys asign the value
values = "one"
my_dict = dict.fromkeys(keys,values)
```

[369]:
```
my_dict
```

[369]:
```
{1: 'one',
 2: 'one',
 3: 'one',
 4: 'one',
 5: 'one',
 6: 'one',
 7: 'one',
 8: 'one',
 9: 'one'}
```

1.2.1 Accessing to value

[370]:
```
dict = {1:"One", 2: 'Two', 3:"Three",4:"Four",5:"Five",6:"Six"}
```

[371]:
```
dict
```

[371]:
```
{1: 'One', 2: 'Two', 3: 'Three', 4: 'Four', 5: 'Five', 6: 'Six'}
```

[372]:
```
dict[2]
```

[372]:
```
'Two'
```

[373]:
```
dict.get(4)
```

[373]:
```
'Four'
```

[376]:
```
myDict ={}
myDict["name"] = 'Darshanikanta'
myDict["designation"] = "AI enginear"
myDict["age"] = 22
myDict["DOB"] = "12/03/2003"
```

[377]:
```
myDict
```

[377]:
```
{'name': 'Darshanikanta',
 'designation': 'AI enginear',
 'age': 22,
```

```
 'DOB': '12/03/2003'}
```

[378]:
```
myDict["name"] = "Darshanikanta behera" # update the value in he dictionary
```

[379]:
```
myDict
```

[379]:
```
{'name': 'Darshanikanta behera',
 'designation': 'AI enginear',
 'age': 22,
 'DOB': '12/03/2003'}
```

[380]:
```
myDict['salary'] = "45k" # add any items into the dictionary
```

[381]:
```
myDict
```

[381]:
```
{'name': 'Darshanikanta behera',
 'designation': 'AI enginear',
 'age': 22,
 'DOB': '12/03/2003',
 'salary': '45k'}
```

[382]:
```
myDict.pop("DOB") # delete out any items through keys
```

[382]:
```
'12/03/2003'
```

[383]:
```
myDict
```

[383]:
```
{'name': 'Darshanikanta behera',
 'designation': 'AI enginear',
 'age': 22,
 'salary': '45k'}
```

[384]:
```
myDict.popitem() # popitem delete last item in the list
```

[384]:
```
('salary', '45k')
```

[385]:
```
myDict
```

[385]:
```
{'name': 'Darshanikanta behera', 'designation': 'AI enginear', 'age': 22}
```

[386]:
```
my_dict
```

[386]:
```
{1: 'one',
 2: 'one',
 3: 'one',
 4: 'one',
 5: 'one',
```

```
 6: 'one',
 7: 'one',
 8: 'one',
 9: 'one'}
```

[253]:
```
my_dict.clear()
```

[254]:
```
my_dict
```

[254]:
```
{}
```

[255]:
```
my_dict = myDict.copy()
```

[256]:
```
myDict
```

[256]:
```
{'name': 'Darshanikanta behera', 'designation': 'AI enginear', 'age': 22}
```

1.2.2 dictionary membership

[387]:
```
'age' in myDict # membership can check usimg the only keys
```

[387]:
```
True
```

[388]:
```
"AI enginear" in myDict # cannot check member using values
```

[388]:
```
False
```

1.2.3 All / Any

[262]:
```
all(myDict)
```

[262]:
```
True
```

[263]:
```
any(myDict)
```

[263]:
```
True
```

2 Tuple , Dictionary Completed