tuple-set-dict-task3-4

August 20, 2024

1 '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

1.1 Tuple Creation

[195]:
```
tup1 = ()
```

[197]:
```
tup2 = (10,30,60)
```

[199]:
```
tup3 = (10,77,30,66,60,89)
```

[201]:
```
tup4 = ('one','two','three')
```

[203]:
```
tup5 = ('Asif',25,(50,100),(150,90))
```

[205]:
```
tup6 = (100,'Asif',17.765)
```

[207]:
```
tup7 =('Asif',25,[50,100],[150,90],{'John','David'},(99,22,33))
```

[209]:
```
len(tup7)
```

[209]:
```
6
```

1.2 Tuple Indexing

[212]:
```
tup2[0]
```

[212]:
```
10
```

[214]:
```
tup4[0]
```

[214]:
```
'one'
```

[216]:
```
tup4[0][0]
```

[216]:
```
'o'
```

[218]:
```
tup4[-1]
```

[218]:
```
'three'
```

[220]:
```
tup5[-1]
```

[220]:
```
(150, 90)
```

1.3 Tuple Slicing

[356]:
```
mytuple = ('one','two','three','four','five','six','seven','eight')
```

[358]:
```
mytuple[0:3]
```

[358]:
```
('one', 'two', 'three')
```

[360]:
```
mytuple[2:5]
```

[360]:
```
('three', 'four', 'five')
```

[362]:
```
mytuple[:3]
```

[362]:
```
('one', 'two', 'three')
```

[364]:
```
mytuple[:2]
```

[364]:
```
('one', 'two')
```

[366]:
```
mytuple[-3]
```

[366]:
```
'six'
```

[368]:
```
mytuple[-2]
```

[368]:
```
'seven'
```

[370]:
```
mytuple[-1]
```

[370]:
```
'eight'
```

[322]:
```
mytuple[:]
```

[322]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

1.4 Remove & Change Items

[373]:
```
mytuple
```

[373]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[375]:
```
del mytuple[0]
```

```
---------------------------------------------------------------
TypeError                         Traceback (most recent call last)
Cell In[375], line 1
----> 1 del mytuple[0]

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

[377]:
```
mytuple[0]=1
```

```
---------------------------------------------------------------
TypeError                         Traceback (most recent call last)
Cell In[377], line 1
----> 1 mytuple[0]=1

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

[379]:
```
del mytuple
```

1.5 Loop trough a tuple

[381]:
```
mytuple = ('one','two','three','four','five','six','seven','eight')
```

[383]:
```
mytuple
```

[383]:
```
('one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight')
```

[385]:
```
for i in mytuple:
    print(i)
```

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

[387]:
```
for i in enumerate(mytuple):
    print(i)
```

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

1.6 Count

[389]:
```
mytuple1 = ('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
```

[391]:
```
mytuple1.count('one')
```

[391]: 3

[393]:
```
mytuple1.count('two')
```

[393]: 2

[395]:
```
mytuple1.count('four')
```

[395]: 1

1.7 Tuple Membership

[397]:
```
mytuple
```

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

[399]:
```
'one' in mytuple
```

[399]: True

[401]:
```
'ten' in mytuple
```

[401]: False

[403]:
```
if 'three' in mytuple:
    print('Three is present in the tuple')
else:
    print('Three is not present in the tuple')
```

```
Three is present in the tuple
```

[405]:
```
if 'eleven' in mytuple:
    print('eleven is present in the tuple')
else:
    print('eleven is not present in the tuple')
```

```
eleven is not present in the tuple
```

1.8 Index Postion

[274]:
```
mytuple
```

[274]:
```
('one', 'two', 'three', 'four', 'four', 'five', 'six', 'seven', 'eight')
```

[276]:
```
mytuple.index('one')
```

[276]:
```
0
```

[278]:
```
mytuple.index('five')
```

[278]:
```
5
```

[407]:
```
mytuple1
```

[407]:
```
('one', 'two', 'three', 'four', 'one', 'one', 'two', 'three')
```

[409]:
```
mytuple1.index('one')
```

[409]:
```
0
```

1.9 Sorting

[412]:
```
mytuple2 = (43,67,99,12,6,90,67)
```

[414]:
```
sorted(mytuple2)
```

[414]:
```
[6, 12, 43, 67, 67, 90, 99]
```

[416]:
```
sorted(mytuple2, reverse=True)
```

[416]:
```
[99, 90, 67, 67, 43, 12, 6]
```

2 Sets

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.

2.1 Set Creation

[418]:
```
myset = {1,2,3,4,5}
myset
```

[418]:
```
{1, 2, 3, 4, 5}
```

[420]:
```
len(myset)
```

[420]:
```
5
```

[422]:
```
my_set = {1,1,2,2,3,4,5,5}
my_set
```

[422]:
```
{1, 2, 3, 4, 5}
```

[424]:
```
myset1 = {1.79,2.08,3.99,4.56,5.45}
myset1
```

[424]:
```
{1.79, 2.08, 3.99, 4.56, 5.45}
```

[426]:
```
myset2 = {'Asif' , 'John' , 'Tyrion'}
myset2
```

[426]:
```
{'Asif', 'John', 'Tyrion'}
```

[428]:
```
myset3 = {10,20, "Hola", (11, 22, 32)}
myset3
```

[428]:
```
{(11, 22, 32), 10, 20, 'Hola'}
```

[430]:
```
myset3 = {10,20, "Hola", [11, 22, 32]}
myset3
```

```
---------------------------------------------------------------
TypeError                         Traceback (most recent call last)
Cell In[430], line 1
----> 1 myset3 = {10,20, "Hola", [11, 22, 32]}
      2 myset3

TypeError: unhashable type: 'list'
```

[432]:
```
myset4 = set()
print(type(myset4))
```

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

[434]:
```
my_set1 = set(('one' , 'two' , 'three' , 'four'))
my_set1
```

[434]:
```
{'four', 'one', 'three', 'two'}
```

2.2 Loop through a Set

[445]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
```

[449]:
```
for i in myset:
    print(i)
```

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

[453]:
```
for i in enumerate(myset):
    print(i)
```

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

2.3 Set Membership

[455]:
```
myset
```

[455]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[457]:
```
'one' in myset
```

[457]:
```
True
```

[459]:
```
'ten' in myset
```

[459]: False

[465]:
```
if 'three' in myset:
    print('Three is present in the set')
else:
    print('Three is not present in the set')
```

Three is present in the set

[469]:
```
if 'eleven' in myset:
    print('eleven is present in the set')
else:
    print('eleven is not present in the set')
```

eleven is not present in the set

2.4 Add & Remove Items

[472]:
```
myset
```

[472]: {'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

[474]:
```
myset.add('NINE')
myset
```

[474]: {'NINE', 'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}

[476]:
```
myset.update(['TEN' , 'ELEVEN' , 'TWELVE'])
myset
```

[476]:
```
{'ELEVEN',
 'NINE',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[478]:
```
myset.remove('NINE')
myset
```

[478]:
```
{'ELEVEN',
 'TEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[480]:
```
myset.discard('TEN')
myset
```

[480]:
```
{'ELEVEN',
 'TWELVE',
 'eight',
 'five',
 'four',
 'one',
 'seven',
 'six',
 'three',
 'two'}
```

[482]:
```
myset.clear()
myset
```

[482]:
```
set()
```

[484]:
```
del myset
myset
```

```
---------------------------------------------------------------
NameError                         Traceback (most recent call last)
Cell In[484], line 2
      1 del myset
----> 2 myset

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

2.5 Copyset

[487]:
```
myset = {'one', 'two', 'three', 'four', 'five', 'six', 'seven', 'eight'}
myset
```

[487]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[489]:
```
myset1 = myset
myset
```

[489]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[493]:
```
id(myset) , id(myset1)
```

[493]:
```
(1891831932608, 1891831932608)
```

[497]:
```
my_set = myset.copy()
my_set
```

[497]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

[499]:
```
id(my_set)
```

[499]:
```
1891831933728
```

[501]:
```
myset.add('nine')
myset
```

[501]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

[503]:
```
myset1
```

[503]:
```
{'eight', 'five', 'four', 'nine', 'one', 'seven', 'six', 'three', 'two'}
```

[505]:
```
my_set
```

[505]:
```
{'eight', 'five', 'four', 'one', 'seven', 'six', 'three', 'two'}
```

2.6 Set Operation

2.6.1 Union

[509]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
C = {8,9,10}
```

[511]:
```
A | B
```

[511]: {1, 2, 3, 4, 5, 6, 7, 8}

[513]:
```
A.union(B)
```

[513]: {1, 2, 3, 4, 5, 6, 7, 8}

[515]:
```
A.union(B, C)
```

[515]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

[517]:
```
"""
Updates the set calling the update() method with union of A , B & C.
For below example Set A will be updated with union of A,B & C.
"""
A.update(B,C)
A
```

[517]: {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}

2.6.2 Intersection

[519]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[521]:
```
A & B
```

[521]: {4, 5}

[523]:
```
A.intersection(B) Intersection of A and B
```

```
  Cell In[523], line 1
    A.intersection(B) Intersection of A and B
                      ^
SyntaxError: invalid syntax
```

[525]:
```
"""
Updates the set calling the intersection_update() method with the intersection o
For below example Set A will be updated with the intersection of A & B.
"""
A.intersection_update(B)
A
```

[525]: {4, 5}

2.6.3 Difference

[527]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[529]:
```
A-B
```

[529]:
```
{1, 2, 3}
```

[531]:
```
A.difference(B)
```

[531]:
```
{1, 2, 3}
```

[533]:
```
B-A
```

[533]:
```
{6, 7, 8}
```

[535]:
```
B.difference(A)
```

[535]:
```
{6, 7, 8}
```

[537]:
```
"""
Updates the set calling the difference_update() method with the difference of se
For below example Set B will be updated with the difference of B & A.
"""
B.difference_update(A)
B
```

[537]:
```
{6, 7, 8}
```

2.6.4 Symmetric Difference

[540]:
```
A = {1,2,3,4,5}
B = {4,5,6,7,8}
```

[542]:
```
A ^ B
```

[542]:
```
{1, 2, 3, 6, 7, 8}
```

[544]:
```
A.symmetric_difference(B)
```

[544]:
```
{1, 2, 3, 6, 7, 8}
```

[546]:
```
"""
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 A & B.
"""
```

```
A.symmetric_difference_update(B)
A
```

[546]: {1, 2, 3, 6, 7, 8}

2.6.5 Subset , Superset & Disjoint

[549]:
```
A = {1,2,3,4,5,6,7,8,9}
B = {3,4,5,6,7,8}
C = {10,20,30,40}
```

[551]:
```
B.issubset(A)
```

[551]: True

[553]:
```
A.issuperset(B)
```

[553]: True

[555]:
```
C.isdisjoint(A)
```

[555]: True

[557]:
```
B.isdisjoint(A)
```

[557]: False

2.6.6 Other Builtin functions

[560]:
```
A
```

[560]: {1, 2, 3, 4, 5, 6, 7, 8, 9}

[562]:
```
sum(A)
```

[562]: 45

[564]:
```
max(A)
```

[564]: 9

[566]:
```
min(A)
```

[566]: 1

[568]:
```
len(A)
```

[568]: 9

[570]:
```
list(enumerate(A))
```

[570]: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 7), (7, 8), (8, 9)]

[572]:
```
D= sorted(A,reverse=True)
D
```

[572]: [9, 8, 7, 6, 5, 4, 3, 2, 1]

[574]:
```
sorted(D)
```

[574]: [1, 2, 3, 4, 5, 6, 7, 8, 9]

3 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

3.1 Create Dictionary

[576]:
```
mydict = dict() # empty dictionary
mydict
```

[576]: {}

[578]:
```
mydict = {} # empty dictionary
mydict
```

[578]: {}

[580]:
```
mydict = {1:'one' , 2:'two' , 3:'three'} # dictionary with integer keys
mydict
```

[580]: {1: 'one', 2: 'two', 3: 'three'}

[582]:
```
mydict = dict({1:'one' , 2:'two' , 3:'three'}) # Create dictionary using dict()
mydict
```

[582]: {1: 'one', 2: 'two', 3: 'three'}

[584]:
```
mydict = {'A':'one' , 'B':'two' , 'C':'three'} # dictionary with character keys
mydict
```

[584]: {'A': 'one', 'B': 'two', 'C': 'three'}

[586]:
```
mydict = {1:'one' , 'A':'two' , 3:'three'} # dictionary with mixed keys
mydict
```

[586]:
```
{1: 'one', 'A': 'two', 3: 'three'}
```

[588]:
```
mydict.keys()
```

[588]:
```
dict_keys([1, 'A', 3])
```

[590]:
```
mydict.values()
```

[590]:
```
dict_values(['one', 'two', 'three'])
```

[592]:
```
mydict.items()
```

[592]:
```
dict_items([(1, 'one'), ('A', 'two'), (3, 'three')])
```

[594]:
```
mydict = {1:'one' , 2:'two' , 'A':['asif' , 'john' , 'Maria']} # dictionary with
mydict
```

[594]:
```
{1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}
```

[600]:
```
keys = {'a' , 'b' , 'c' , 'd'}
mydict3 = dict.fromkeys(keys)
mydict3
```

[600]:
```
{'b': None, 'a': None, 'd': None, 'c': None}
```

[602]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = 10
mydict3 = dict.fromkeys(keys , value)
mydict3
```

[602]:
```
{'b': 10, 'a': 10, 'd': 10, 'c': 10}
```

[604]:
```
keys = {'a' , 'b' , 'c' , 'd'}
value = [10,20,30]
mydict3 = dict.fromkeys(keys , value)
mydict
```

[604]:
```
{1: 'one', 2: 'two', 'A': ['asif', 'john', 'Maria']}
```

[606]:
```
value.append(40)
mydict3
```

[606]:
```
{'b': [10, 20, 30, 40],
 'a': [10, 20, 30, 40],
```

```
 'd': [10, 20, 30, 40],
 'c': [10, 20, 30, 40]}
```

3.1.1 Accessing Items

[609]:
```
mydict = {1:'one' , 2:'two' , 3:'three' , 4:'four'}
mydict
```

[609]:
```
{1: 'one', 2: 'two', 3: 'three', 4: 'four'}
```

[611]:
```
mydict[1]
```

[611]:
```
'one'
```

[613]:
```
mydict.get(1)
```

[613]:
```
'one'
```

[615]:
```
mydict1 = {'Name':'Asif' , 'ID': 74123 , 'DOB': 1991 , 'job' :'Analyst'}
mydict1
```

[615]:
```
{'Name': 'Asif', 'ID': 74123, 'DOB': 1991, 'job': 'Analyst'}
```

[617]:
```
mydict1['Name']
```

[617]:
```
'Asif'
```

[619]:
```
mydict1.get('job')
```

[619]:
```
'Analyst'
```

3.1.2 Add, Remove & Change Items

[622]:
```
mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict1
```

[622]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

[624]:
```
mydict1['DOB'] = 1992 # Changing Dictionary Items
mydict1['Address'] = 'Delhi'
mydict1
```

[624]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1992, 'Address': 'Delhi'}
```

[626]:
```
dict1 = {'DOB':1995}
mydict1.update(dict1)
mydict1
```

[626]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}
```

[628]:
```
mydict1['Job'] = 'Analyst' # Adding items in the dictionary
mydict1
```

[628]:
```
{'Name': 'Asif',
 'ID': 12345,
 'DOB': 1995,
 'Address': 'Delhi',
 'Job': 'Analyst'}
```

[630]:
```
mydict1.pop('Job')
mydict1
```

[630]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995, 'Address': 'Delhi'}
```

[632]:
```
mydict1.popitem()
```

[632]:
```
('Address', 'Delhi')
```

[634]:
```
mydict1
```

[634]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1995}
```

[636]:
```
del[mydict1['ID']]
mydict1
```

[636]:
```
{'Name': 'Asif', 'DOB': 1995}
```

[638]:
```
mydict1.clear()
mydict1
```

[638]:
```
{}
```

[640]:
```
del mydict1
mydict1
```

```
---------------------------------------------------------------
NameError                         Traceback (most recent call last)
Cell In[640], line 2
      1 del mydict1
----> 2 mydict1

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

3.1.3 Copy Dictionary

[643]:
```
mydict = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki'}
mydict
```

[643]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

[645]:
```
mydict1 = mydict
```

[647]:
```
id(mydict) , id(mydict1)
```

[647]:
```
(1891833605952, 1891833605952)
```

[649]:
```
mydict2 = mydict.copy()
```

[651]:
```
id(mydict2)
```

[651]:
```
1891829586176
```

[653]:
```
mydict['Address'] = 'Mumbai'
```

[655]:
```
mydict
```

[655]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```

[657]:
```
mydict1
```

[657]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Mumbai'}
```

[659]:
```
mydict2
```

[659]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

3.1.4 Loop through a Dictionary

[664]:
```
mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Address' : 'Hilsinki' }
mydict1
```

[664]:
```
{'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Address': 'Hilsinki'}
```

[666]:
```
for i in mydict1:
    print(i , ':' , mydict1[i])
```

```
Name : Asif
ID : 12345
DOB : 1991
Address : Hilsinki
```

[668]:
```
for i in mydict1:
    print(mydict1[i])
```

```
Asif
12345
1991
Hilsinki
```

[670]:
```
mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
mydict1
```

[670]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}

[672]:
```
'Name' in mydict1
```

[672]: True

[674]:
```
'Asif' in mydict1
```

[674]: False

[676]:
```
'ID' in mydict1
```

[676]: True

[678]:
```
'Address' in mydict1
```

[678]: False

[680]:
```
mydict1 = {'Name':'Asif' , 'ID': 12345 , 'DOB': 1991 , 'Job': 'Analyst'}
mydict1
```

[680]: {'Name': 'Asif', 'ID': 12345, 'DOB': 1991, 'Job': 'Analyst'}

[682]:
```
all(mydict1)
```

[682]: True

[ ]:
```

```