06. Tuple & Set

September 17, 2025

1 Tuple & Set

Note Book Owner: Emdadul Hoque

1.1 Creating Tuples

[1]:
```
t1 = () #create empty tuples
t2 = (12, 13, 14, 15) #create a typlw with 4 numeric element
t3 = ("Emdadul", "Hoque", "Tareque") #Create a tuple with string
t4 = "a", "b", "c", "d" # create a typle without parenthesis
```

[4]:
```
t = t1, t2, t3, t4
```

[12]:
```
t
```

[12]:
```
((), (12, 13, 14, 15), ('Emdadul', 'Hoque', 'Tareque'), ('a', 'b', 'c', 'd'))
```

[15]:
```
# Point to remember: A single value in parenthesis is not a tuple

t5 = (4)
print("Single vaule in parenthesis", type(t5))
t6 = (4, )
print("Single vaule in parenthesis with comma", type(t6))
```

```
Single vaule in parenthesis <class 'int'>
Single vaule in parenthesis with comma <class 'tuple'>
```

[6]:
```
lst1 = []
lst2 = [1, 2, 3]
lst3 = ["a", "b"]
```

[7]:
```
lst = lst1, lst2, lst3
```

[8]:
```
lst
```

[8]:
```
([], [1, 2, 3], ['a', 'b'])
```

[11]:
```
type(lst)
```

[11]:
```
tuple
```

[20]:
```
### Create a tuple with tuple  function

string = "Emdadul"
t1 = tuple(string)
```

[21]:
```
t1
```

[21]:
```
('E', 'm', 'd', 'a', 'd', 'u', 'l')
```

[25]:
```
# only one argument accept tuple
t2 = tuple(["Emdadul", 1, 2, 3], [1, 2, 3, 4])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[25], line 1
----> 1 t2 = tuple(["Emdadul", 1, 2, 3], [1, 2, 3, 4])

TypeError: tuple expected at most 1 argument, got 2
```

[26]:
```
t2 = (["Emdadul", 1, 2, 3], [1, 2, 3, 4])
```

[27]:
```
t2
```

[27]:
```
(['Emdadul', 1, 2, 3], [1, 2, 3, 4])
```

1.2 Inbuilt functions for tuples

len()–> Returns the number of elements in a tuple max()–> Returns the element with the greatest value min()–> Returns the element with the smallest value sum()–> Returns the sum of all the elements of a tuple index(x)–> Returns the index of element x count(x)–> Returns the number of occurrences of element x

[30]:
```
t2 = (1, 2, 3, 4)
```

[31]:
```
sum(t2)
```

[31]:
```
10
```

[39]:
```
t3 = ([1,2,3,4],"a","b","c", ("Emdadul", "Haque", 25, 27))
```

[40]:
```
t3
```

[40]:
```
([1, 2, 3, 4], 'a', 'b', 'c', ('Emdadul', 'Haque', 25, 27))
```

[41]:
```
len(t3)
```

[41]: 5

[51]:
```
t4 = (1, 2, 3, 4,100, -100, 20, 30,3,4,5)
```

[43]:
```
max(t4)
```

[43]: 100

[44]:
```
min(t4)
```

[44]: -100

[46]:
```
t4.index(-100)
```

[46]: 5

[61]:
```
t4.count(3)
```

[61]: 2

1.3 Tuple indexing & slicing

[193]:
```
t5 = (1, 2, 3, 4, 5, 6)
```

[65]:
```
t5[4], t5[3]
```

[65]: (5, 4)

[55]:
```
t5[::-1]
```

[55]: (6, 5, 4, 3, 2, 1)

[56]:
```
t5
```

[56]: (1, 2, 3, 4, 5, 6)

[59]:
```
t5[::-1]
```

[59]: (6, 5, 4, 3, 2, 1)

[60]:
```
t5[::2]
```

[60]: (1, 3, 5)

[67]:
```
t5[2:4]
```

[67]: (3, 4)

1.3.1 Operations of tuples

[68]:
```
# The + Operator
t1 = (1, 2, 3)
t2 = (4, 5, 6)
```

[70]:
```
t1 = t1+t2
```

[71]:
```
t1
```

[71]:
```
(1, 2, 3, 4, 5, 6)
```

[74]:
```
t1+(" ", )
```

[74]:
```
(1, 2, 3, 4, 5, 6, ' ')
```

[75]:
```
# The * operator
t3 = (2, 3, 4)
t3 = t3*3
```

[76]:
```
t3
```

[76]:
```
(2, 3, 4, 2, 3, 4, 2, 3, 4)
```

[1]:
```
t = ((1, 2, 3, 4))

for i in t:
    print(i)
```

```
1
2
3
4
```

[87]:
```
#traverse tuples from a list
t = [(1,"Emdadul"), (2, "Hoque"), (3, "Tareque")]
for no, name in t:
    print(no, name)
```

```
1 Emdadul
2 Hoque
3 Tareque
```

1.4 The zip() Function

[109]:
```
lst1 = [1, 2, 3, 4]
lst2 = "abcd"
```

[110]:
```
list(zip(lst1, lst2))
```

[110]:
```
[(1, 'a'), (2, 'b'), (3, 'c'), (4, 'd')]
```

[112]:
```
# another example

lst1 =["Laptop", "Mobile", "Headphone"]
lst2 = [100000, 12000, 5000]
```

[113]:
```
lst3 = list(zip(lst1, lst2))
```

[114]:
```
lst3
```

[114]:
```
[('Laptop', 100000), ('Mobile', 12000), ('Headphone', 5000)]
```

[115]:
```
tpl = tuple(lst3)
```

[116]:
```
tpl
```

[116]:
```
(('Laptop', 100000), ('Mobile', 12000), ('Headphone', 5000))
```

[117]:
```
tple = tuple(zip(lst1, lst2))
```

[126]:
```
tple
```

[126]:
```
(('Laptop', 100000), ('Mobile', 12000), ('Headphone', 5000))
```

1.4.1 The inverse zip(*) function

[124]:
```
product, prize = zip(*tple)
print(product)
print(prize)
```

```
('Laptop', 'Mobile', 'Headphone')
(100000, 12000, 5000)
```

[125]:
```
type(product)
```

[125]:
```
tuple
```

[127]:
```
#more examples on zip (*) function

matrix = [(1,2), (3, 4), (5,6)]

z = zip(*matrix)
```

[129]:
```
tuple(z)
```

[129]:
```
((1, 3, 5), (2, 4, 6))
```

[131]:
```
matrix = [[1, 2], [3, 4], [5, 6]]

z = zip(*matrix)
```

[135]:
```
tuple(z)
```

[135]:
```
()
```

Program 03: Consider an example of a tuple as T = (1, 3, 2, 4, 6, 5). Write a program to store numbers present at odd index into a new tuple.

[136]:
```
def odd_tuple(tple):
    return tple[::2]

tple = (1, 2, 3, 4, 5, 6)
new_tple = odd_tuple(tple)

print(new_tple)
```

```
(1, 3, 5)
```

[134]:
```
new_tple
```

[134]:
```
(1, 3, 5)
```

2 Sets

2.1 Creating Sets

[143]:
```
s1 = {1, 2, 3, 4, 5}
```

[144]:
```
s1
```

[144]:
```
{1, 2, 3, 4, 5}
```

[146]:
```
#if you want to create a empty, you can not you empty curly brackets
s2 = {}
```

[147]:
```
type(s2)
```

[147]:
```
dict
```

[150]:
```
# if you want to create empty set, using in build set function
s1 = set()
```

[151]:
```
s1
```

[151]:
```
set()
```

[152]:
```
#convert list to set
lst = [1, 2, 3, 4 , 5, 6]
s = set(lst)
```

[153]:
```
s
```

[153]:
```
{1, 2, 3, 4, 5, 6}
```

[154]:
```
# why we use set: when you want to enique value in your list

lst = [1, 2, 3, 4, 5, 2, 2, 3, 1, 4, 3]

s = set(lst)
```

[155]:
```
s
```

[155]:
```
{1, 2, 3, 4, 5}
```

[173]:
```
# the set in and not in operator
s = {1, 2, 3, 4}
3 in s
```

[173]:
```
True
```

[175]:
```
4 not in s
```

[175]:
```
False
```

2.1.1 zip() function and zip(*) function in set

[169]:
```
s1 = {1, 2, 3, 4, 5, 6}
s2 = {10, 20, 30, 40, 50, 60}
```

[170]:
```
s3 = set(zip(s1, s2))
```

[170]:
```
{(1, 50), (2, 20), (3, 40), (4, 10), (5, 60), (6, 30)}
```

2.2 Methods of Set class

[176]:
```
s = {10, 20, 30, 40, 50, 60}
```

[177]:
```
s.add(70) # add a element
```

[178]:
```
s
```

[178]:
```
{10, 20, 30, 40, 50, 60, 70}
```

[180]:
```
s.remove(30) # remove a element from a set
```

[181]:
```
s
```

[181]: {10, 20, 40, 50, 60, 70}

[182]:
```
s.clear() # remove all element
```

[183]:
```
s
```

[183]: set()

[184]:
```
s1 = {1, 2, 3, 4}
s2 = {1, 2, 3, 4, 5}

s1.issubset(s2)
```

[184]: True

[197]:
```
s2.issuperset(s1)
```

[197]: False

[199]:
```
s = {2, 3, 1, 4, 5, 6}
```

[204]:
```
s1 = sorted(s)
```

[205]:
```
print(s1)
print(type(s1))
```

```
[1, 2, 3, 4, 5, 6]
<class 'list'>
```

2.3 Set operations

[186]:
```
s1 = {1, 2, 3, 4, 5}
s2 = {2, 4, 5, 6}
```

[187]:
```
s1.union(s2)
```

[187]: {1, 2, 3, 4, 5, 6}

[188]:
```
s1.intersection(s2)
```

[188]: {2, 4, 5}

[189]:
```
s1.difference(s2)
```

[189]: {1, 3}

[190]:
```
s1.symmetric_difference(s2)
```

```
[190]: {1, 3, 6}
```