07-Python Basic

Python Tuple and Set data types

Posted on September 11, 2021
Last updated on February 2, 2023

7.1 Tuple Data Structure

Tuple is exactly same as List except that it is immutable. i.e once we creates Tuple object,we cannot perform any changes in that object. Hence Tuple is `Read Only version of List.`

- Tuple are `immutable` and `insertion order is preserved`
- `Duplicates` and `Heterogeneous` objectsare allowed

```
# Def empty tuple
p = ()
p = tuple()

t=(10)   # int
t=(10,)  # tuple
t=(10,20,30,40)  # tuple

# Convert into tuples
tuple([1,2,3])
# (1,2,3)

tuple("Amrit")
# ('A', 'm', 'r', 'i', 't')
```

```
t4 = ((1,2), 3, 'ap' , [7,8])
t4[-1][0] = 6
print(t4)
# ((1, 2), 3, 'ap', [6, 8])
```

7.1.1 Tuple Packing and Unpacking

```
# packing
p = 1,2,3,4
# (1, 2, 3, 4)

# unpacking
t = (10,20,30,40)
a, b, c, d = t
print(a,b,c,d)
# 10 20 30 40
```

At the time of tuple unpacking the number of variables and number of values should be same, otherwise we will get ValueError.

7.1.3 Mathematical operators for tuple

- Concatenation Operator(+)

```
t1=(10,20,30)
t2=(40,50,60)

t3=t1+t2
print(t3)
# (10,20,30,40,50,60)
```

- Multiplication operator or repetition operator(*)

```
t1=(10,20,30)
t2=t1*3
print(t2)
# (10,20,30,10,20,30,10,20,30)
```

7.1.4 Tuple in-built function

- `len()`, `count()`. `index()`

```
t1 = (True, 'sudh', (45+7j), 45.45, 45, 4, 3, 2, 1)

# True is also 1
print(t1.index(1))  # 0

# 1 is also True
print(t1.count(True))  # 2
```

- `min()` and `max()`

```
t=(40,10,30,20)
print(min(t))  # 10
print(max(t))  # 40
```

- `sorted()`

```
t=(40,10,30,20)
t1=sorted(t)
print(t1)  # [10, 20, 30, 40]
print(t)   # (40, 10, 30, 20)
```

- `cmp()`
  - It compares the elements of both tuples.

- If both tuples are equal then returns 0
- If the first tuple is less than second tuple then it returns -1
- If the first tuple is greater than second tuple then it returns +1
- Is available `only in Python2` but not in Python 3

```
t1=(10,20,30)
t2=(40,50,60)
t3=(10,20,30)

print(cmp(t1,t2))  # -1
print(cmp(t1,t3))  # 0
print(cmp(t2,t3))  # +1
```

7.1.5 Tuple Comprehension

- Tuple Comprehension is not supported by Python.
- Here we are not getting tuple object and we are `getting generator object`.

```
t = ( x**2 for x in range(1,6))
print(type(t))  # <class 'generator'>

for x in t:
    print(x, end=' ')
# 1 4 9 16 25
```

7.2 Set Data Structure

If we want to represent a group of `unique values` as a single entity then we should go for set.

- `Duplicates`, `Indexing` and `slicing` are **not allowed**.
- Insertion order is not preserved.But we can sort the elements.
- `Heterogeneous` elements are allowed.
- Set objects are `mutable`
- Can apply mathematical operations like `union`, `intersection`, `difference` etc on set objects.

```
# Def empty set
p = set()

# `Not set but dict`
p2 = {}

{1,2,'a'}


# Element contains in set must be `immutable type only`
p.add([1,2])  # --> Error
p.add((1,2))  # --> Ok
```

Mathematical operations on the Set

- union()
  - We can use this function to return all elements present in both sets

```
# x.union(y) or x|y

x={10,20,30,40}
y={30,40,50,60}

print(x.union(y))
# {10, 20, 30, 40, 50, 60}

print(x|y)
# {10, 20, 30, 40, 50, 60}
```

- intersection()
  - Returns common elements present in both x and y

```
# x.intersection(y) or x&y

x={10,20,30,40}
y={30,40,50,60}

print(x.intersection(y))
# {40, 30}

print(x&y)
# {40, 30}
```

- `difference()`
    - Returns the elements present in x but not in y

```
# x.difference(y) or x-y

x={10,20,30,40}
y={30,40,50,60}

print(x.difference(y))
# {10, 20}

print(x-y)
# {10, 20}

print(y-x)
# {50, 60}
```

Set in-built function

- `pop()`

```
s={40,10,30,20}
print(s.pop()) # 40
print(s) # {10, 20, 30}
```

- `remove(x)`, `discard(x)`

```
s = {40,10,30,20}
s.remove(30)
print(s)
# {40, 10, 20}

# KeyError: 50
s.remove(50)

# No Error
s.discard(50)
```

- `add(x)`
  - Adds item x to the set

```
s = {10,20,30}
s.add(40);
print(s)
# {40, 10, 20, 30}
```

- `update(x, y, z)`
  - To add multiple items to the set

```
s = {10, 20}
s.update([40, 50], (1, 2))
print(s)
# {1, 2, 40, 10, 50, 20}
```

Set Comprehension

```
input_list = [1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 7]

set_using_comp = {var for var in input_list if var % 2 == 0}
print(set_using_comp)
# {2, 4, 6}
```

.

.