9 Tuples

"Ordered, heterogenous, immutable...."

Contents

What are Tuples?

- Though a list can store dissimilar data, it is commonly used for storing similar data.
- Though a tuple can store similar data it is commonly used for storing dissimilar data. The tuple data is enclosed within ( ) as shown below.

```
a = ( )                          # empty tuple
b = (10,)                        # tuple with one item. , after 10 is necessary
c = ('Sanjay', 25, 34555.50)     # tuple with dissimilar items
d = (10, 20, 30, 40)             # tuple with similar items
```

While creating the tuple **b**, if we do not use the comma after 10, **b** is treated to be of type **int**.

- While initializing a tuple, we may drop ( ).

```
c = 'Sanjay', 25, 34555.50       # tuple with multiple items
print(type(c))                   # c is of the type tuple
```

- Items in a tuple can be repeated, i.e. tuple may contain duplicate items. However, unlike list, tuple elements cannot be repeated using a *.

```
tpl1 = (10,) * 5                 # stores (10, 10, 10, 10, 10)
tpl2 = (10) * 5                  # stores (50)
```

Accessing Tuple Elements

- Entire tuple can be printed by just using the name of the tuple.

```
tpl = ('Sanjay', 25, 34555.50)
print(tpl)
```

- Tuple is an ordered collection. So order of insertion of elements in a tuple is same as the order of access. So like a string and list, tuple items too can be accessed using indices, starting with 0.

```
msg = ('Handle', 'Exceptions', 'Like', 'a', 'boss')
print(msg[1], msg[3])
```

- Like strings and lists, tuples too can be sliced to yield smaller tuples.

```
emp = ('Sanjay', 23, 23000, 1760, 2040)
print(emp[1:3])          # prints (23, 23000)
print(emp[3:])           # prints  (1760, 2040)
print(emp[:3])           # prints ('Sanjay', 23, 23000)
```

Looping in Tuples

- If we wish to process each item in a tuple, we should be able to iterate through it. This can be done using a while loop or **for** loop.

```
tpl = (10, 20, 30, 40, 50)
i = 0
while i < len(tpl) :
   print(tpl[i])
   i += 1
for n in tpl :
     print(n)
```

- While iterating through a tuple using a **for** loop, if we wish to keep track of index of the element that is being currently processed, we can do so using the built-in **enumerate( )** function.

```
tpl = (10, 20, 30, 40, 50)
for index, n in enumerate(tpl) :
     print(index, n)
```

Basic Tuple Operations

- Mutability - Unlike a list, a tuple is immutable, i.e. it cannot be modified.

```
msg = ('Fall', 'In', 'Line')
msg[0] ='FALL'                    # error
msg[1:3] = ('Above', 'Mark')      # error
```

- Since a tuple is immutable operations like append, remove and insert do not work with a tuple.

- Though a tuple itself is immutable, it can contain mutable objects like lists.

```
# mutable lists, immutable string—all can belong to tuple
s = ([1, 2, 3, 4], [4, 5], 'Ocelot')
```

- If a tuple contains a list, the list can be modified since list is a mutable object.

```
s = ([1, 2, 3, 4], [10, 20], 'Oynx')
s[1][1] = 45      # changes first item of first list, i.e. 20
print(s)          # prints ([1, 2, 3, 4], [4, 45], 'Oynx')

# one more way to change first item of first list
p = s[1]
p[1] = 100
print(s)          # prints ([1, 2, 3, 4], [4, 100], 'Oynx')
```

- The other basic operations that are done on a tuple are very similar to the ones done on a list. These operations are discussed in Chapter 8. You may try the following operations on tuples as an exercise:

  Concatenation
  Merging
  Conversion
  Aliasing
  Cloning
  Searching
  Identity
  Comparison
  Emptiness

Using Built-in Functions on Tuples

- Many built-in functions can be used with tuples.

```
t = (12, 15, 13, 23, 22, 16, 17)   # create tuple
len(t)          # return number of items in tuple t
max(t)          # return maximum element in tuple t
min(t)          # return minimum element in tuple t
sum(t)          # return sum of all elements in tuple t
any(t)          # return True if any element of tpl is True
all(t)          # return True if all elements of tpl are True
sorted(t)       # return sorted list (not sorted tuple)
reversed(t)     # used for reversing t
```

Tuple Methods

- Any tuple is an object of type **tuple**. Its methods can be accessed using the syntax **tpl.method( )**. Usage of two methods is shown below:

```
tpl = (12, 15, 13, 23, 22)     # create tuple
print(tpl.count(23))           # return no. of times 23 appears in lst
print(tpl.index(22))           # return index of item 22
print(tpl.index(50))           # reports valueError as 50 is absent in lst
```

Tuple Varieties

- It is possible to create a tuple of tuples.

```
a = (1, 3, 5, 7, 9)
b = (2, 4, 6, 8, 10)
c = (a, b)
print(c[0][0], c[1][2])  # 0th element of 0th tuple, 2nd ele of 1st tuple
```

```
records = (
              ('Priyanka', 24, 3455.50), ('Shailesh', 25, 4555.50),
              ('Subhash', 25, 4505.50), ('Sugandh', 27, 4455.55)
           )
print(records[0][0], records[0][1], records[0][2])
print(records[1][0], records[1][1], records[1][2])
for n, a, s in records :
    print(n,a,s)
```

- A tuple may be embedded in another tuple.

```
x = (1, 2, 3, 4)
y = (10, 20, x, 30)
print(y)                    # outputs (10, 20, (1, 2, 3, 4), 30)
```

- It is possible to unpack a tuple within a tuple using the *operator.

```
x = (1, 2, 3, 4)
y = (10, 20, *x, 30)
print(y)                    # outputs (10, 20, 1, 2, 3, 4, 30)
```

- It is possible to create a list of tuples, or a tuple of lists.

```
lst = [('Priyanka', 24, 3455.50), ('Shailesh', 25, 4555.50)]
tpl = (['Priyanka', 24, 3455.50], ['Shailesh', 25, 4555.50])
```

- If we wish to sort a list of tuples or tuple of lists, it can be done as follows:

```
import operator
# each embedded tuple/list contains name, age, salary
lst = [('Shailesh', 24, 3455.50), ('Priyanka', 25, 4555.50)]
tpl = (['Shailesh', 24, 3455.50], ['Priyanka', 25, 4555.50])
print(sorted(lst))
print(sorted(tpl))
print(sorted(lst, key = operator.itemgetter(2)))
print(sorted(tpl, key = operator.itemgetter(2)))
```

- By default, **sorted( )** sorts by first item in list/tuple, i.e. name.
- If we wish to sort by salary, we need to use the **itemgetter( )** function of **operator** module.
- The **key** parameter of **sorted( )** requires a key function (to be applied to objects to be sorted) rather than a single key value.
- **operator.itemgetter(2)** will give us a function that fetches salary from a list/tuple.
- In general, **operator.itemgetter(n)** constructs a function that takes a list/tuple as input, and fetches the n-th element out of it.

P</> Programs

Problem 8.1

Pass a tuple to the **divmod( )** function and obtain the quotient and the remainder.

Program

```
result = divmod(17,3)
print(result)
t = (17, 3)
result = divmod(*t)
```

```
print(result)
```

Output

```
(5, 2)
(5, 2)
```

Tips

- If we pass **t** to **divmod( )** an error is reported. We have to unpack the tuple into two distinct values and then pass them to **divmod( )**.
- **divmod( )** returns a tuple consisting of quotient and remainder.

Problem 8.2

Write a Python program to perform the following operations:

- Pack first 10 multiples of 10 into a tuple
- Unpack the tuple into 10 variables, each holding 1 value
- Unpack the tuple such that first value gets stored in variable x, last value in y and all values in between into disposable variables _
- Unpack the tuple such that first value gets stored in variable i, last value in j and all values in between into a single disposable variable _

Program

```
tpl = (10, 20, 30, 40, 50, 60, 70, 8, 90, 100)
a, b, c, d, e, f, g, h, i, j = tpl
print(tpl)
print(a, b, c, d, e, f, g, h, i, j)
x, _, _, _, _, _, _, _, _, y = tpl
print(x, y, _)
i, *_, j = tpl
print(i, j, _)
```

Output

```
(10, 20, 30, 40, 50, 60, 70, 8, 90, 100)
10 20 30 40 50 60 70 8 90 100
10 100 90
10 100 [20, 30, 40, 50, 60, 70, 8, 90]
```

Tips

- Disposable variable _ is usally used when you do not wish to use the variable further, and is being used only as a place-holder.

Problem 8.3

A list contains names of boys and girls as its elements. Boys' names are stored as tuples. Write a Python program to find out number of boys and girls in the list.

Program

```
lst = ['Shubha', 'Nisha', 'Sudha', ('Suresh',), ('Rajesh',), 'Radha']
boys = 0
girls = 0
for ele in lst:
   if isinstance(ele, tuple):
      boys += 1
   else :
      girls += 1
print('Boys = ', boys, 'Girls = ', girls)
```

Output

```
Boys =  2 Girls =  4
```

Tips

- **isinstance( )** functions checks whether **ele** is an instance of tuple type.
- Note that since the tuples contain a single element, it is followed by a comma.

Problem 8.4

A list contains tuples containing roll number, names and age of student. Write a Python program to gather all the names from this list into another list.

Program

```
lst = [('A101', 'Shubha', 23), ('A104', 'Nisha', 25), ('A111', 'Sudha', 24)]
nlst = [ ]
for ele in lst:
   nlst = nlst + [ele[1]]

print(nlst)
```

Output

```
['Shubha', 'Nisha', 'Sudha']
```

Tips

- **nlst** is an empty to begin with. During each iteration name is extracted from the tuple using **ele[1**] and added to the current list of names in **nlst**.

Problem 8.5

Given the following tuple

('F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd')

Write a Python program to carry out the following operations:

- Add an ! at the end of the tuple
- Convert a tuple to a string
- Extract ('b', 'b') from the tuple
- Find out number of occurrences of 'e' in the tuple
- Check whether 'r' exists in the tuple
- Convert the tuple to a list
- Delete characters 'b, 'b', 'e', 'r' from the tuple

Program

```
tpl = ('F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd')

# addition of ! is not possible as tuple is an immutable
# so to add ! we need to create a new tuple and then make tpl refer to it
tpl = tpl + ('!',)
print(tpl)
```

```
# convert tuple to string
s = ''.join(tpl)
print(s)

# extract ('b', 'b') from the tuple
t = tpl[3:5]
print(t)

# count number of 'e' in the tuple
count = tpl.count('e')
print('count = ', count)

# check whether 'r' exists in the tuple
print('r' in tpl)

# Convert the tuple to a list
lst = list(tpl)
print(lst)

# tuples are immutable, so we cannot remove elements from it
# we need to split the tuple, eliminate the unwanted element and then merge the tuples
tpl = tpl[:3] + tpl[7:]
print(tpl)
```

Output

```
('F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd', '!')
Flabbergasted!
('b', 'b')
count =  2
True
['F', 'l', 'a', 'b', 'b', 'e', 'r', 'g', 'a', 's', 't', 'e', 'd', '!']
('F', 'l', 'a', 'g', 'a', 's', 't', 'e', 'd', '!')
```

[A] Which of the following properties apply to string, list and tuple?

- Iterable
- Sliceable
- Indexable
- Immutable
- Sequence
- Can be empty
- Sorted collection
- Ordered collection
- Unordered collection
- Elements can be accessed using their position in the collection

[B] Which of the following operations can be performed on string, list and tuple?

- a = b + c
- a += b
- Appending a new element at the end
- Deletion of an element at the 0th position
- Modification of last element
- In place reversal

[C] Answer the following questions:

(a) Is this a valid tuple?

```
tpl = ('Square')
```

(b) What will be the output of the following code snippet?

```
num1 = num2 = (10, 20, 30, 40, 50)
print(id(num1), type(num2))
print(isinstance(num1, tuple))
print(num1 is num2)
print(num1 is not num2)
print(20 in num1)
print(30 not in num2)
```

(c) Suppose a date is represented as a tuple (d, m, y). Write a program to create two date tuples and find the number of days between the two dates.

(d) Create a list of tuples. Each tuple should contain an item and its price in float. Write a program to sort the tuples in descending order by price. Hint: Use **operator.itemgetter( )**.

(e) Store the data about shares held by a user as tuples containing the following information about shares:

Share name
Date of purchase
Cost price
Number of shares
Selling price

Write a program to determine:

- Total cost of the portfolio.
- Total amount gained or lost.
- Percentage profit made or loss incurred.

(f) Write a program to remove empty tuple from a list of tuples.

(g) Write a program to create following 3 lists:

- a list of names
- a list of roll numbers
- a list of marks

Generate and print a list of tuples containing name, roll number and marks from the 3 lists. From this list generate 3 tuples—one containing all names, another containing all roll numbers and third containing all marks.

[D] Match the following pairs:

a. tpl1 = ('A',)	1. tuple of length 6
b. tpl1 = ('A')	2. tuple of lists
c. t = tpl[::-1]	3. Tuple
d. ('A', 'B', 'C', 'D')	4. list of tuples
e. [(1, 2), (2, 3), (4, 5)]	5. String
f. tpl = tuple(range(2, 5))	6. Sorts tuple
g. ([1, 2], [3, 4], [5, 6])	7. (2, 3, 4)
h. t = tuple('Ajooba')	8. tuple of strings
i. [*a, *b, *c]	9. Unpacking of tuples in a list
j. (*a, *b, *c)	10. Unpacking of lists in a tuple