Tuples in Python

Estimated time needed: **15** minutes

Objectives

After completing this lab you will be able to:

- Perform the basics tuple operations in Python, including indexing, slicing and sorting

Imagine you received album recommendations from your friends and compiled all of the recommendations into a table, with specific information about each album.

The table has one row for each movie and several columns:

- **artist** - Name of the artist
- **album** - Name of the album
- **released_year** - Year the album was released
- **length_min_sec** - Length of the album (hours,minutes,seconds)
- **genre** - Genre of the album
- **music_recording_sales_millions** - Music recording sales (millions in USD) on SONG://DATABASE
- **claimed_sales_millions** - Album's claimed sales (millions in USD) on SONG://DATABASE
- **date_released** - Date on which the album was released
- **soundtrack** - Indicates if the album is the movie soundtrack (Y) or (N)
- **rating_of_friends** - Indicates the rating from your friends from 1 to 10

The dataset can be seen below:

Artist Album Released Length Genre Music recording sales (millions) Claimed sales (millions) Released Soundtrack Rating (friends) Michael Jackson Thriller 1982 00:42:19 Pop, rock, R&B 46 65 30-Nov-82 10.0 AC/DC Back in Black 1980 00:42:11 Hard rock 26.1 50 25-Jul-80 8.5 Pink Floyd The Dark Side of the Moon 1973 00:42:49 Progressive rock 24.2 45 01-Mar-73 9.5 Whitney Houston The Bodyguard 1992 00:57:44 Soundtrack/R&B, soul, pop 26.1 50 25-Jul-80 Y 7.0 Meat Loaf Bat Out of Hell 1977 00:46:33 Hard rock, progressive rock 20.6 43 21-Oct-77 7.0 Eagles Their Greatest Hits (1971-1975) 1976 00:43:08 Rock, soft rock, folk rock 32.2 42 17-Feb-76 9.5 Bee Gees Saturday Night Fever 1977 1:15:54 Disco 20.6 40 15-Nov-77 Y 9.0 Fleetwood Mac Rumours 1977 00:40:01 Soft rock 27.9 40 04-Feb-77 9.5

In Python, there are different data types: string, integer, and float. These data types can all be contained in a tuple as follows:

Now, let us create your first tuple with string, integer and float.

```
# Create your first tuple

tuple1 = ("disco",10,1.2 )
tuple1
```

The type of variable is a **tuple**.

```
# Print the type of the tuple you created

type(tuple1)
```

Each element of a tuple can be accessed via an index. The following table represents the relationship between the index and the items in the tuple. Each element can be obtained by the name of the tuple followed by a square bracket with the index number:

We can print out each value in the tuple:

```
# Print the variable on each index

print(tuple1[0])
print(tuple1[1])
print(tuple1[2])
```

We can print out the **type** of each value in the tuple:

```
# Print the type of value on each index

print(type(tuple1[0]))
print(type(tuple1[1]))
print(type(tuple1[2]))
```

We can also use negative indexing. We use the same table above with corresponding negative values:

We can obtain the last element as follows (this time we will not use the print statement to display the values):

```
# Use negative index to get the value of the last element

tuple1[-1]
```

We can display the next two elements as follows:

```
# Use negative index to get the value of the second last element

tuple1[-2]

# Use negative index to get the value of the third last element

tuple1[-3]
```

We can concatenate or combine tuples by using the **+** sign:

```
# Concatenate two tuples

tuple2 = tuple1 + ("hard rock", 10)
tuple2
```

We can slice tuples obtaining multiple values as demonstrated by the figure below:

We can slice tuples, obtaining new tuples with the corresponding elements:

```
# Slice from index 0 to index 2

tuple2[0:3]
```

We can obtain the last two elements of the tuple:

```
# Slice from index 3 to index 4

tuple2[3:5]
```

We can obtain the length of a tuple using the length command:

```
# Get the length of tuple

len(tuple2)
```

This figure shows the number of elements:

Consider the following tuple:

```
# A sample tuple

Ratings = (0, 9, 6, 5, 10, 8, 9, 6, 2)
```

We can sort the values in a tuple and save it to a new tuple:

```
# Sort the tuple

RatingsSorted = sorted(Ratings)
RatingsSorted
```

A tuple can contain another tuple as well as other more complex data types. This process is called 'nesting'. Consider the following tuple with several elements:

```
# Create a nest tuple

NestedT =(1, 2, ("pop", "rock") ,(3,4),("disco",(1,2)))
```

Each element in the tuple, including other tuples, can be obtained via an index as shown in the figure:

```
# Print element on each index

print("Element 0 of Tuple: ", NestedT[0])
print("Element 1 of Tuple: ", NestedT[1])
print("Element 2 of Tuple: ", NestedT[2])
print("Element 3 of Tuple: ", NestedT[3])
print("Element 4 of Tuple: ", NestedT[4])
```

We can use the second index to access other tuples as demonstrated in the figure:

We can access the nested tuples:

```
# Print element on each index, including nest indexes

print("Element 2, 0 of Tuple: ",   NestedT[2][0])
print("Element 2, 1 of Tuple: ",   NestedT[2][1])
print("Element 3, 0 of Tuple: ",   NestedT[3][0])
print("Element 3, 1 of Tuple: ",   NestedT[3][1])
print("Element 4, 0 of Tuple: ",   NestedT[4][0])
print("Element 4, 1 of Tuple: ",   NestedT[4][1])
```

We can access strings in the second nested tuples using a third index:

```
# Print the first element in the second nested tuples

NestedT[2][1][0]

# Print the second element in the second nested tuples

NestedT[2][1][1]
```

We can use a tree to visualise the process. Each new index corresponds to a deeper level in the tree:

Similarly, we can access elements nested deeper in the tree with a third index:

```
# Print the first element in the second nested tuples

NestedT[4][1][0]

# Print the second element in the second nested tuples

NestedT[4][1][1]
```

The following figure shows the relationship of the tree and the element NestedT[4][1][1]:

Consider the following tuple:

```
# sample tuple

genres_tuple = ("pop", "rock", "soul", "hard rock", "soft rock", \
                "R&B", "progressive rock", "disco")
genres_tuple
```

Find the length of the tuple, genres_tuple:

```
# Write your code below and press Shift+Enter to execute
len(genres_tuple)
```

Access the element, with respect to index 3:

```
# Write your code below and press Shift+Enter to execute
genres_tuple[3]
```

Use slicing to obtain indexes 3, 4 and 5:

```
# Write your code below and press Shift+Enter to execute
genres_tuple[3:6]
```

Find the first two elements of the tuple genres_tuple:

```
# Write your code below and press Shift+Enter to execute
genres_tuple[0:2]
```

Find the first index of "disco":

```
# Write your code below and press Shift+Enter to execute
genres_tuple.index("disco")
```

Generate a sorted List from the Tuple C_tuple=(-5, 1, -3):

```
# Write your code below and press Shift+Enter to execute
C_tuple = (-5, 1, -3)
C_list = sorted(C_tuple)
C_list
```

Author

Joseph Santarcangelo

Other contributors

Mavis Zhou

Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2022-01-10	2.1	Malika	Removed the readme for GitShare
2020-08-26	2.0	Lavanya	Moved lab to course repo in GitLab