Tuples in Python

Welcome! This notebook will teach you about the tuples in the Python Programming Language. By the end of this lab, you'll know the basics tuple operations in Python, including indexing, slicing and sorting.

Table of Contents

Estimated time needed: **15 mins**

About the Dataset

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_IDENTIFIER
- **claimed_sales_millions** - Album's claimed sales (millions in USD) on SONG_IDENTIFIER
- **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_year	length_min_sec	genre	music_recording_sales_millions	claimed_sales_millions	date_released	soundtrack	rating_of_friends
Michael Jackson	Thriller	1982	42:19	Pop, rock, R&B	26.9to40.9	50+	30-Nov-82	10.1	NaN
AC/DC	Back in Black	1980	38:42	Hard rock	26.1	50.1	25-Jul-80	8.5	Y
Pink Floyd	The Dark Side of the Moon	1973	42:49	Progressive rock, pop	24.1	45.1	1-Mar-73	9.1	Y
Whitney Houston	The Bodyguard	1992	57:31	Soundtrack/R&B, soul, pop	26.1	50.1	25-Jul-92	7.1	Y
Meat Loaf	Bat Out of Hell	1977	46:30	Hard rock, progressive rock	24.6	43.6	21-Oct-77	7.5	Y
Eagles	Their Greatest Hits (1971-1975)	1976	00:45:28	Rock, soft rock, folk rock	32.2	65.2	17-Feb-76	8.6	Y
Bee Gees	Saturday Night Fever	1977	1:15:54	Disco	26.1	50.1	15-Nov-77	9.6	Y
Fleetwood Mac	Rumours	1977	00:39:35	Soft rock	22.1	45.1	04-Feb-77	9.3	Y
Eagles	Hotel California	1976	00:43:11	Rock, soft rock, folk rock	32.1	65.1	22-Feb-76	9.6	Y
Winger	Winger	1988	00:41:48	Hard rock, soft rock, folk rock	15.1	30.1	09-Aug-88	7.1	N
Soundtrack	Saturday Night Fever	1977	1:15:54	Disco	26.1	50.1	15-Nov-77	9.6	Y
Fleetwood Mac	Rumours	1977	00:39:35	Soft rock	22.1	45.1	04-Feb-77	9.3	Y
Eagles	Hotel California	1976	00:43:11	Rock, soft rock, folk rock	32.1	65.1	22-Feb-76	9.6	Y
Winger	Winger	1988	00:41:48	Hard rock, soft rock, folk rock	15.1	30.1	09-Aug-88	7.1	N

Tuples

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.

In [ ]:
```
# Create your first tuple
tuple1 = ("disco",10,1.2 )
tuple1
```

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

In [ ]:
```
# Print the type of the tuple you created
type(tuple1)
```

Indexing

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:

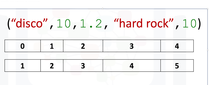
We can print out each value in the tuple:

In [ ]:
```
# Print the variables on each index

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

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

In [ ]:
```
# 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):

In [ ]:
```
# Use negative index to get the value of the last element

tuple1[-1]
```

We can display the next two elements as follows:

In [ ]:
```
# Use negative index to get the value of the second last element

tuple1[-2]
```

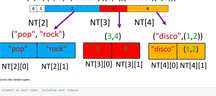
In [ ]:
```
# Use negative index to get the value of the third last element

tuple1[-3]
```

Concatenate Tuples

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

In [ ]:
```
# Concatenate two tuples

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

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



Slicing

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

In [ ]:
```
# Slice from index 0 to index 2

tuple2[0:3]
```

We can obtain the last two elements of the tuple:

In [ ]:
```
# Slice from index 3 to index 4

tuple2[3:5]
```

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

In [ ]:
```
# Get the length of tuple

len(tuple2)
```

This figure shows the number of elements:

Sorting

Consider the following tuple:

In [ ]:
```
# A random 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:

In [ ]:
```
# Sort the tuple

Ratings1=sorted(Ratings)
Ratings1
```

Nested Tuples

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:

In [ ]:
```
# Create a nest tuple

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

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

In [ ]:
```
# 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 contained in the figure:

We can access the nested tuples :

In [ ]:
```
# 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:

In [ ]:
```
# Print the first element in the second nested tuples
NestedT[2][1][0]
```

In [ ]:
```
# 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 fourth index:

In [ ]:
```
# Print the first element in the second nested tuples

NestedT[2][1][0]
```

In [ ]:
```
# Print the second element in the second nested tuples

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

The following figure shows the relationship of the tree and the element `NestedT[2][1][1]`:

Quiz on Tuples

Consider the following tuple:

In [ ]:
```
# Genres 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`:

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

Double click **here** for the solution.

Access the element, with respect to index 3:

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

Double click **here** for the solution.

Use slicing to obtain indices 3, 4 and 5:

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

Double click **here** for the solution.

Find the first two elements of the tuple, `genres_tuple`:

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

Double click **here** for the solution.

Find the first index of `"disco"`:

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

Double click **here** for the solution.

Generate a sorted List from the Tuple `C_tuple=(-5, 1, -15)`:

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

Double click **here** for the solution.

The last exercise!

Congratulations, you have completed your first lesson and hands-on lab in Python. However, there is one more thing you need to do. The Data Science community encourages sharing work. The best way to share and showcase your work is to share it on GitHub. By sharing your notebook on GitHub you are not only building your reputation with fellow data scientists, but you can show off your capacity to code. Even though the rate you find pictures of work, it never has early to start building good habits. So, please read and follow our advice to learn how to share your work.

About the Authors:

Joseph Santarcangelo is a Data Scientist at IBM, and holds a PhD in Electrical Engineering. His research focused on using Machine Learning, Signal Processing, and Computer Vision to determine how videos impact human cognition. Joseph has been working for IBM since he completed his PhD.

Other contributors: Mavis Zhou