quiz-tuple

September 16, 2024

0.1 QUIZ - Tuple

Q 1:

Define a function named **create_tuple**.

It will create a tuple of numbers from 1 to 10.

Then it will add numbers from 11 to 20 to this tuple.

And it will return the final form of the tuple.

Hints: * Tuples are Immutable. So how do you add elements to a Tuple? * use `for` loop * `range()`

[1]:
```
# Q 1:

# ---- your solution here ----
def create_tuple():
    initial_tuple = tuple(range(1, 11))

    for i in range(11, 21):
        initial_tuple += (i,)

    return initial_tuple
# call the function you defined
t = create_tuple()
print(t)
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20)
```

Q 2:

Define a function named **tuple_to_string**.

It will take a Tuple as parameter and convert it into a String.

Then it will return this String.

Hints: * `join()`

[2]:
```
# Q 2:
```

```
# ---- your solution here ----
def tuple_to_string(t):
    return ''.join(t)

# call the function you defined
t = ('M', 'a', 'c', 'h', 'i', 'n', 'e', ' ', 'L', 'e', 'a', 'r', 'n', 'i', 'n',␣
 ↪'g')
t = tuple_to_string(t)
print(t)
```

```
Machine Learning
```

Q 3:

Define a function named **string_to_list_to_tuple**.

It will take String as parameter.

First it will convert this String into a List, then it will convert this List into a Tuple.

Finally it will return the Tuple.

Hints: * only use `constructor` methods * list() * tuple()

[3]:
```
# Q 3:

# ---- your solution here ----
def string_to_list_to_tuple(s):
    return tuple(list(s))

# call the function you defined
pythons_father = 'Guido van Rossum'
father_tuple = string_to_list_to_tuple(pythons_father)
print(father_tuple)
```

```
('G', 'u', 'i', 'd', 'o', ' ', 'v', 'a', 'n', ' ', 'R', 'o', 's', 's', 'u', 'm')
```

Q 4:

Define a function named **how_many_instances**.

It will take a Tuple and an index (int) as parameter.

Function will first find the element at that index.

Then it will return the number of instances of the element in the Tuple.

Hints: * count()

[4]:
```
# Q 4:

# ---- your solution here ----
def how_many_instances(t, i):
    element = t[i]
    return t.count(element)

# call the function you defined
t = (5, 2, 3, 3, 4, 2, 1, 3, 4, 5, 2, 1, 2)
i = 1
count = how_many_instances(t, i)
print(count)
```

```
4
```

Q 5:

Slicing Operations in Tuples are exactly the same as in Lists and Strings.

Our Tuple is:

Find the items in tup, based on slicing and indexing: 1. items from position 4 (inc) to position 7 (inc) 2. the first 5 items 3. items from position 6 (inc) 4. all items 5. 2nd item from the last 6. last 4 items 7. items from position 2 to 8, with step size 2 8. all items with step size 3 9. items from position 9 to position 3 (exc) but in reverse order 10. print the tup in reverse order 11. print the tup in reverse order with step size 2 12. all the elements excluding the last one, with negative index

[5]:
```
tup = ('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j')

# 1. items from position 4 (inc) to position 7 (inc)
result_1 = tup[3:7]

# 2. the first 5 items
result_2 = tup[:5]

# 3. items from position 6 (inc)
result_3 = tup[5:]

# 4. all items
result_4 = tup[:]

# 5. 2nd item from the last
result_5 = tup[-2]

# 6. last 4 items
result_6 = tup[-4:]
```

```python
# 7. items from position 2 to 8, with step size 2
result_7 = tup[1:8:2]

# 8. all items with step size 3
result_8 = tup[::3]

# 9. items from position 9 to position 3 (exc) but in reverse order
result_9 = tup[8:2:-1]

# 10. print the tup in reverse order
result_10 = tup[::-1]

# 11. print the tup in reverse order with step size 2
result_11 = tup[::-2]

# 12. all the elements excluding the last one, with negative index
result_12 = tup[:-1]

print(result_1)
print(result_2)
print(result_3)
print(result_4)
print(result_5)
print(result_6)
print(result_7)
print(result_8)
print(result_9)
print(result_10)
print(result_11)
print(result_12)
```

```
('d', 'e', 'f', 'g')
('a', 'b', 'c', 'd', 'e')
('f', 'g', 'h', 'i', 'j')
('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j')
i
('g', 'h', 'i', 'j')
('b', 'd', 'f', 'h')
('a', 'd', 'g', 'j')
('i', 'h', 'g', 'f', 'e', 'd')
('j', 'i', 'h', 'g', 'f', 'e', 'd', 'c', 'b', 'a')
('j', 'h', 'f', 'd', 'b')
('a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i')
```

Q 6:

Define a function named **change_tuple_ending**.

It will take a List as parameter.

The elements of this list will be Tuples.

Each Tuple may be in different length, so the size of the elements in the list is not fixed.

The function will replace the last item in each Tuple element in the list with its square.

Hints: * Mutate the original list (the parameter list will change in-place) * How can you get last element in a Tuple? * How to mutate the list, while looping over it? * You need to find a way to mutate the list (enumerate()) * How to create a Tuple with single element?

[6]:
```
# Q 6:

# ---- your solution here ----


# call the function you defined
tuple_list = [(2,5,8), (4,3), (1,7,9,6), (5,)]
print('tuple_list before passing to function:', tuple_list)

change_tuple_ending(tuple_list)
print('tuple_list after passing to function:', tuple_list)
```

```
tuple_list before passing to function: [(2, 5, 8), (4, 3), (1, 7, 9, 6), (5,)]
tuple_list after passing to function: [(2, 5, 64), (4, 9), (1, 7, 9, 36), (25,)]
```

Q 7:

Define a function named **replace_tuple_with_square**.

It will take a List as parameter.

The items in this List are Tuples.

Each Tuple may be in different length, so the size of the elements in the list is not fixed.

The function will replace each element of Tuples with its square.

And it will return the new list of squared Tuples.

Hint: * Do not change the parameter * How to mutate the list while looping over it?

[9]:
```
# Q 7:

# ---- your solution here ----
def replace_tuple_with_square(tuple_list):
    return [tuple(x**2 for x in tpl) for tpl in tuple_list]

# call the function you defined
tuple_list = [(2,5,8), (4,3), (1,7,9,6), (5,)]
```

```
new_tuple_list = replace_tuple_with_square(tuple_list)

print('tuple_list:', tuple_list)
print('new_tuple_list:', new_tuple_list)
```

```
tuple_list: [(2, 5, 8), (4, 3), (1, 7, 9, 6), (5,)]
new_tuple_list: [(4, 25, 64), (16, 9), (1, 49, 81, 36), (25,)]
```

Q 8:

Define a function named **movie_characters**.

It will take 4 lists as parameters.

4 lists: * List 1 -> Actor/Actress Name * List 2 -> Movie Title * List 3 -> Release Year * List 4 -> Character Name

Function will take these 4 lists and create 2 Tuples out of them.

Each Tuple will have 2 elements. * Tuple 1 -> (Actor/Actress Name, Character Name) * Tuple 2 -> (Movie Title, Release Year)

It will use the Tuple 1 as the Key and Tuple 2 as the Value and create a Dictionary.

And it will return this dictionary.

Hints: * `zip()`

[11]:
```
# Q 8:

# ---- your solution here ----
def movie_characters(actors, characters, movies, years):
    tuple1 = list(zip(actors, characters))
    tuple2 = list(zip(movies, years))
    return dict(zip(tuple1, tuple2))

# call the function you defined
# lists:z
actors = ['Marlon Brando', 'Heath Ledger', 'Natalie Portman', 'Emma Stone']
characters = ['Don Vito Corleone', 'Joker', 'The Swan Queen', 'Mia']
movies = ['The Godfather', 'The Dark Knight', 'Black Swan', 'La La Land']
years = [1972, 2008, 2010, 2016]

# call the functions
movie_dictionary = movie_characters(actors, characters, movies, years)
movie_dictionary
```

[11]:
```
{('Marlon Brando', 'Don Vito Corleone'): ('The Godfather', 1972),
 ('Heath Ledger', 'Joker'): ('The Dark Knight', 2008),
 ('Natalie Portman', 'The Swan Queen'): ('Black Swan', 2010),
```

```
('Emma Stone', 'Mia'): ('La La Land', 2016)}
```