

Data structures in Python- tuples

By

Prof. Muhammad Iqbal Bhat

Department of Higher Education
Government Degree College Beerwah

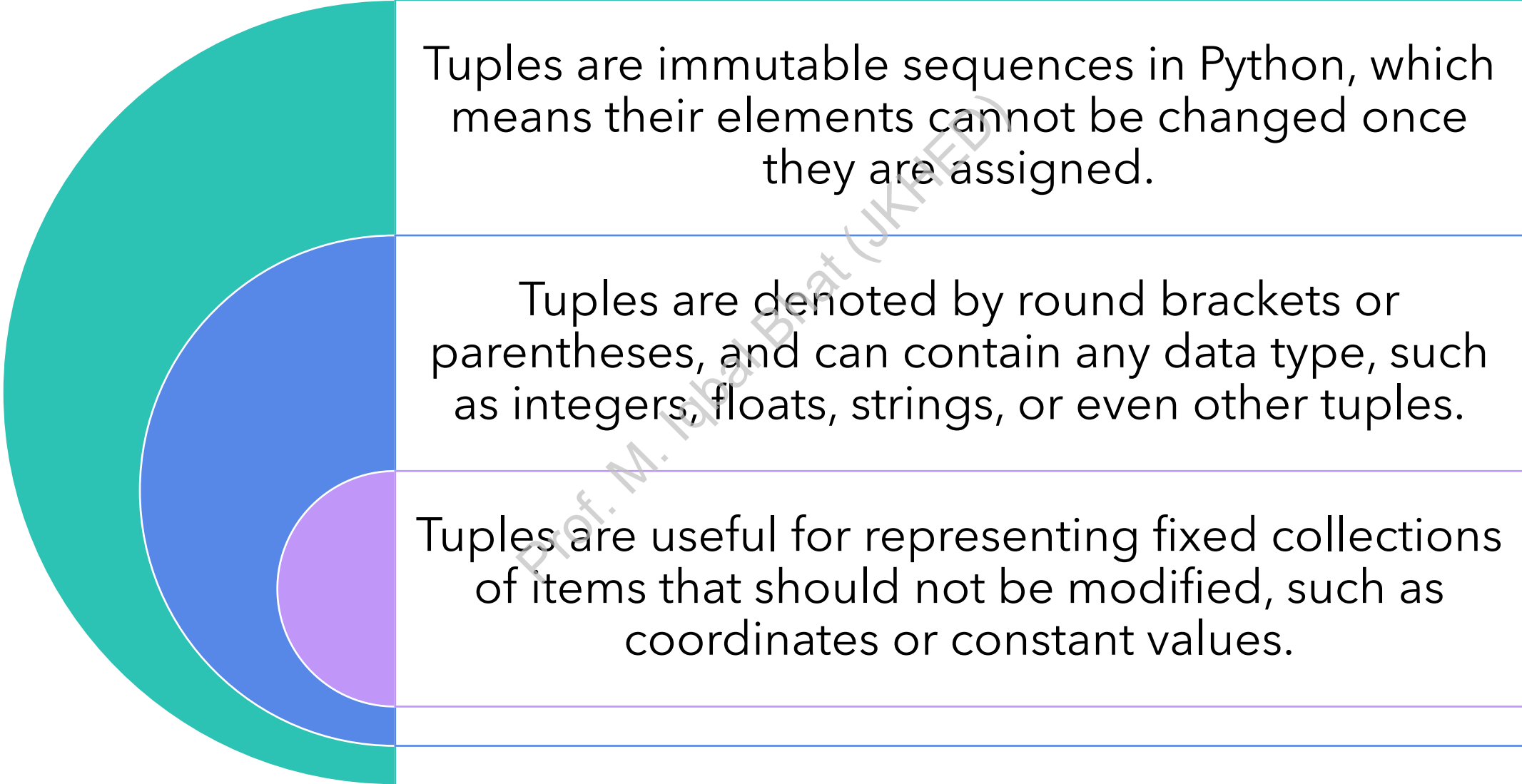
Topics



Python Tuples

Prof. M. Anbal Bhat (JKHED)

(tuples,) in Python:



Tuples are immutable sequences in Python, which means their elements cannot be changed once they are assigned.

Tuples are denoted by round brackets or parentheses, and can contain any data type, such as integers, floats, strings, or even other tuples.

Tuples are useful for representing fixed collections of items that should not be modified, such as coordinates or constant values.

Creating (Tuples):



Tuples can be created using parentheses, and elements are separated by commas.



Example: **my_tuple = (1, 2, 3, 4, 5)**



To create an empty tuple, use empty parentheses



Example: **student_tuple = ()**



can pack a tuple by separating its values with commas



Example: **student_tuple = 'John', 'Green', 3.3**



The following code creates a one-element tuple:



Example: **a_singleton_tuple = ('red',)**

Accessing tuple elements:

Tuple elements can be accessed using indexing, starting from 0 for the first element.

Example:

```
time_tuple = (9, 16, 1)
```

```
time_tuple[0] * 3600 + time_tuple[1] * 60 +
```

```
time_tuple[2]
```

(Tuples):

- Tuple elements can be sliced using the colon operator, which allows us to extract a portion of the tuple.
 - Example: **my_tuple = (1, 2, 3, 4, 5)**
 - **slice_tuple = my_tuple[1:4]**
- Tuples can be concatenated using the operator, which creates a new tuple by combining two or more tuples.
 - Example: **tuple1 = (1, 2, 3)**
 - **tuple2 = (4, 5, 6)**
 - **concatenated_tuple = tuple1 + tuple2**
- Tuple elements can be unpacked into individual variables, which allows us to assign each element to a separate variable.
 - **my_tuple = (1, 2, 3)**
 - **a, b, c = my_tuple**
- You can use += to append a tuple to a list:
 - **numbers = [1, 2, 3, 4, 5]**
 - **numbers += (6, 7)**

tuple Methods:



`count(x)`: Returns the number of occurrences of the specified value `x` in the tuple.



`index(x)`: Returns the index of the first occurrence of the specified value `x` in the tuple.

Prof. M. Iqbal Bhat (JKHED)

Program examples:

```
my_tuple = (1, 2, 3, 4, 2, 2, 5)
```

```
# count()
```

```
count = my_tuple.count(2)
```

```
print(count) # Output: 3
```

```
# index()
```

```
index = my_tuple.index(4)
```

```
print(index) # Output: 3
```

Prof. M. Iqbal Bhat (JKHED)

Prof. M. Iqbal Bhat (JKHED)

Uses of Tuples

Multiple Return Values from a Function

```
def get_name_and_age():  
    name = "John"  
    age = 30  
    return name, age  
  
# Call the function and unpack the returned tuple  
name, age = get_name_and_age()  
  
print("Name:", name)    # Output: Name: John  
print("Age:", age)      # Output: Age: 30
```

Unpacking of Tuple Elements

```
# Define a tuple  
point = (3, 4)
```

```
# Unpack tuple elements into separate variables  
x, y = point
```

```
print("x:", x)    # Output: x: 3  
print("y:", y)    # Output: y: 4
```

Immutable Collections

```
# Define a tuple to represent RGB color  
rgb_color = (255, 0, 0)
```

```
# Attempt to modify tuple elements (will raise  
an error)
```

```
rgb_color[1] = 128 # Raises TypeError: 'tuple'  
object does not support item assignment
```

Prof. M. Jugal Bhat (JKHED)

Dictionary Keys

```
# Define a dictionary with tuples as keys
person_ages = {"John", "Doe": 30, ("Jane",
"Smith"): 25}

# Access values using tuple keys
print(person_ages[("John", "Doe")]) # Output:
30
```

Conclusion:

- Lists are a versatile and powerful data structure in Python.
- Python provides several built-in methods for manipulating lists that can help simplify common list operations.
- Familiarizing yourself with list methods can greatly improve your productivity and effectiveness when working with lists in Python.