Python Datatypes – **Tuple-String- Dictionary- Set**

TUYEN NGOC LE

Outline

- **Tuple**
- **String**
- **Dictionary**
- **Set**

Create a Python Tuple

A tuple in Python is similar to a list. The difference between the two is that we cannot change the elements of a tuple once it is assigned whereas we can change the elements of a list.

A tuple is created by placing all the items (elements) inside parentheses (), separated by commas. The **brackets are optional**, however, it is a good practice to use them.

```
# Different types of tuples
# Empty tuple
my_tuple = () # brackets
print(my_tuple)
# Tuple having integers
my_tuple = (1, 2, 3)
print(my_tuple)
# tuple with mixed datatypes
my_tuple = (1, "Hello", 3.4)
print(my_tuple)
# nested tuple
my_tuple = "mouse", [8, 4, 6], (1, 2, 3)
print(my_tuple)
```

```
# Different types of lists
# Empty tuple
my_list = [] # square brackets
print(my_list)
# Tuple having integers
my_list = [1, 2, 3]
print(my_list)
# tuple with mixed datatypes
my_list = [1, "Hello", 3.4]
print(my_list)
# nested tuple
my_list = ["mouse", [8, 4, 6], (1, 2, 3)]
print(my_list)
```

Create a Python Tuple With one Element

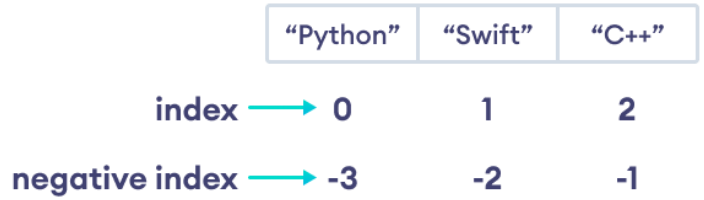
In Python, creating a tuple with one element is a bit tricky. Having one element within parentheses is not enough. We will need a **trailing comma** to indicate that it is a tuple

```
var1 = ("hello")
print(type(var1))  # <class 'str'>
# Creating a tuple having one element
var2 = ("hello", ) # trailing comma
print(type(var2))  # <class 'tuple'>
# Parentheses is optional
var3 = "hello",    # trailing comma
print(type(var3))  # <class 'tuple'>
```

Access Python Tuple Elements

Like a list, each element of a tuple is represented by index numbers or use the for loop to iterate over the elements of a tuple

```
letters = ("p", "r", "o", "g", "r", "a", "m", "i", "z")
# accessing tuple elements using indexing
print(letters[0])   # prints "p"
print(letters[-1])   # prints "a"
# accessing tuple elements using slicing
# elements 2nd to 4th index
print(my_tuple[1:4])  #  prints('r', 'o', 'g')
# elements beginning to 2nd
print(my_tuple[:-7]) # prints('p', 'r')
# elements 8th to end
# elements beginning to end
print(my_tuple[:]) # Prints('p', 'r', 'o', 'g', 'r', 'a', 'm',
'i', 'z')
# iterating through the tuple
for letter in letters :
print(letter)
```



Python Tuple Methods

Method	Description
count()	Returns the number of times a specified value occurs in a tuple
index()	Searches the tuple for a specified value and returns the position of where it was found

```
#Return the number of times the value 5 appears in the tuple
thistuple = (1, 3, 7, 8, 7, 5, 4, 6, 8, 5)
x = thistuple.count(0)
print(x)
# Search for the first occurrence of the value 8, and return its position
y = thistuple.index(8)
print(y)
# Check if an Item Exists in the Python Tuple
print('0' in thistuple)
```

Advantages of Tuple over List in Python

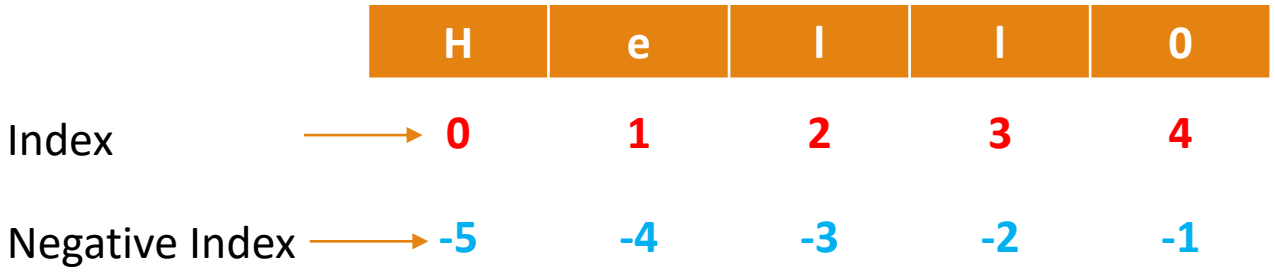
- Since tuples are quite similar to lists, both of them are used in similar situations.
- However, there are certain advantages of implementing a tuple over a list:
- We generally use tuples for heterogeneous (different) data types and lists for homogeneous (similar) data types.
- Since tuples are immutable, iterating through a tuple is faster than with a list. So there is a slight performance boost.
- Tuples that contain immutable elements can be used as a key for a dictionary. With lists, this is not possible.
- If you have data that doesn't change, implementing it as tuple will guarantee that it remains write-protected.

Python Strings

- In computer programming, a string is a sequence of characters. For example, "hello" is a string containing a sequence of characters 'h', 'e', 'l', 'l', and 'o'.
- We use single quotes '' or double quotes "" to represent a string in Python

```
# create string type variables
name = "Python"
print(name)
message = "I love Python."
print(message)
# multiline string
message = """
Never gonna give you up
Never gonna let you down
"""
print(message)
```

Access String Characters in Python




```
greet = 'hello'
# access 1st index element
print(greet[1]) # "e"
# access 4th last element
print(greet[-4]) # "e"
greet = 'Hello'
# access character from 1st index to 3rd index
print(greet[1:4])  # "ell"
# iterating through greet string
for letter in greet:
     print(letter)
```

Python String Operations

1. Compare Two Strings
2. Join Two or More Strings
3. Python String Length
4. String Membership Test

```
str1 = "Hello, world!"
str2 = "I love Python."
str3 = "Hello, world!"

# compare str1 and str2
print(str1 == str2) # False
# compare str1 and str3
print(str1 == str3) # True
# using + operator
str4 = str1 + str2 + str3
print(str4)
# count length of str4 string
print(len(greet))
# test if a substring exists within a string or not,
print('a' in str4) # False
print('llo' not in str4) #True
```

Methods of Python String

Methods	Description
upper()	converts the string to uppercase
lower()	converts the string to lowercase
partition()	returns a tuple
replace()	replaces substring inside
find()	returns the index of first occurrence of substring
rstrip()	removes trailing characters
split()	splits string from left
startswith()	checks if string starts with the specified string
isnumeric()	checks numeric characters
index()	returns index of substring

Methods of Python String

```
imagepath = 'E:/Datasets/Blood Cell/BCCD/BCCD/ori/Platelets/Platelets_BloodImage_00003_00016.jpg'
print(imagepath.split('/'))
```

split() splits a string at the specified separator and returns a list of substrings.

```
['E:', 'Datasets', 'Blood Cell', 'BCCD', 'BCCD', 'ori', 'Platelets', 'Platelets_BloodImage_00003_00016.jpg']
```

Python Dictionary

- Python dictionary is an ordered collection (starting from **Python 3.7**) of items. It stores elements in **key/value** pairs. Here, **keys** are unique identifiers that are associated with each **value**.
- The values in dictionary items can be of any data type:
- A dictionary is a collection which is ordered (version 3.7, dictionaries are *ordered*. In Python 3.6 and earlier, dictionaries are *unordered*), changeable and do not allow duplicates.

```
car_dict = {
  "brand": "Ford",
  "electric" : False,
  "year" : 1964,
  "colors" : ["red", "white", "blue"] }

print(car_dict)
```

Create a dictionary in Python

```
capital_city = {
"Nepal": "Kathmandu",
"Italy" : "Rome",
"England" : "London" }
print(capital_city)
```

Keys	Values
Nepal	Kathmandu
Italy	Rome
England	London

```
# Create a dictionary
person1_dict = { "name": "John", "age" : 36, "country" : "Norway" }
print(person1_dict)
# use the dict() constructor
person_dict = dict(name = "John", age = 36, country = "Norway")
print(person_dict)
```

Methods for Working with Python Dictionaries

Function	Description
all()	Return True if all keys of the dictionary are True (or if the dictionary is empty).
any()	Return True if any key of the dictionary is true. If the dictionary is empty, return False.
len()	Return the length (the number of items) in the dictionary.
sorted()	Return a new sorted list of keys in the dictionary.
clear()	Removes all items from the dictionary.
keys()	Returns a new object of the dictionary's keys.
values()	Returns a new object of the dictionary's values

Methods for Working with Python Dictionaries

```
capital_city = { "Nepal": "Kathmandu", "Italy" : "Rome", "England" : "London" }
print(capital_city)
# Use the get() Method
Italy_cap = capital_city.get("Italy")
print(Italy_cap)
# Use the items() Dictionary Method
items = capital_city.items()
print(items)
# Use the keys() Dictionary Method
dict_keys = capital_city.keys()
print(dict_keys)
# Use the values() Dictionary Method
dict_values = capital_city.values()
print(dict_values)
```

Python Sets

- A set is a collection of unique data. That is, elements of a set cannot be duplicate.
- Suppose we want to store information about student IDs. Since student IDs cannot be duplicate, we can use a set.
- In Python, we create sets by placing all the elements inside curly braces {}, separated by comma.
- A set can have any number of items and they may be of different types (integer, float, tuple, string etc.). But a set cannot have mutable elements like lists, sets or dictionaries as its elements.

```
# create a set of integer type
student_id = { 112, 114, 116, 118, 115 }
print('Student ID:', student_id)

# create a set of string type
vowel_letters = { 'a', 'e', 'i', 'o', 'u' }
print('Vowel Letters:', vowel_letters)

# create a set of mixed data types
mixed_set = { 'Hello', 101, -2, 'Bye' }
print('Set of mixed data types:', mixed_set)
```

Create an Empty Set in Python

```
# create an empty set
empty_set = set()

# create an empty dictionary
empty_dictionary = { }

# check data type of empty_set
print('Data type of empty_set:', type(empty_set))

# check data type of dictionary_set
print('Data type of empty_dictionary', type(empty_dictionary))
```

Built-in Functions with Set

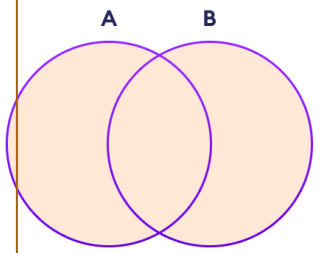
Function	Description
all()	Returns True if all elements of the set are true (or if the set is empty).
any()	Returns True if any element of the set is true. If the set is empty, returns False.
enumerate()	Returns an enumerate object. It contains the index and value for all the items of the set as a pair.
len()	Returns the length (the number of items) in the set.
max()	Returns the largest item in the set.
min()	Returns the smallest item in the set.
sorted()	Returns a new sorted list from elements in the set(does not sort the set itself).
sum()	Returns the sum of all elements in the set.

Built-in Functions with Set

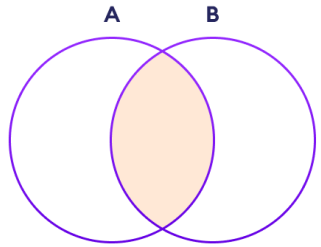
```
# Duplicate Items in a Set
numbers = { 2, 4, 6, 6, 2, 8 }
print(numbers)   # {8, 2, 4, 6}
# Add Items to a Set in Python using add() method
numbers.add(32)
print('Updated Set:', numbers)
# update the set with items other collection types(lists, tuples, sets, etc).
new_number = [-3, -10, -7, 0]
numbers.update(new_number)
print('Updated Set:', numbers)
# use the discard() method to remove the specified element from a set
removed_number = numbers.discard(0)
print('After remove 0: ', numbers)
# Iterate Over a Set in Python
for number in numbers :
print(number)
# find number of elements
print('Total Elements:', len(numbers))
```

Python Set Operations

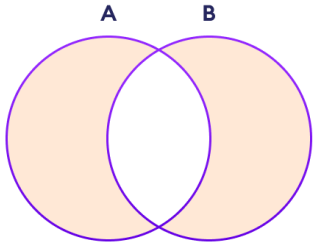
```
# first set
A = { 1, 3, 5 }
# second set
B = { 0, 2, 4 }
# perform union operation using |
print('Union using |:', A | B)
# perform union operation using union()
print('Union using union():', A.union(B))
# perform intersection operation using&
print('Intersection using &:', A& B)
# perform intersection operation using intersection()
print('Intersection using intersection():', A.intersection(B))
# perform difference operation using&
print('Difference using &:', A - B)
# perform difference operation using difference()
print('Difference using difference():', A.difference(B))
# perform difference operation using&
print('using ^:', A^ B)
# using symmetric_difference()
print('using symmetric_difference():',
A.symmetric_difference(B))
```



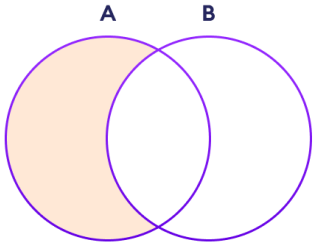
Set Union



Set Intersection



Set Difference



Set Difference

Python Set Methods

Method	Description
add()	Adds an element to the set
clear()	Removes all elements from the set
copy()	Returns a copy of the set
difference()	Returns the difference of two or more sets as a new set
discard()	Removes an element from the set if it is a member. (Do nothing if the element is not in set)
intersection()	Returns the intersection of two sets as a new set
intersection_update()	Updates the set with the intersection of itself and another
isdisjoint()	Returns True if two sets have a null intersection
issubset()	Returns True if another set contains this set
issuperset()	Returns True if this set contains another set
pop()	Removes and returns an arbitrary set element. Raises KeyError if the set is empty
remove()	Removes an element from the set. If the element is not a member, raises a KeyError
symmetric_difference()	Returns the symmetric difference of two sets as a new set
union()	Returns the union of sets in a new set
update()	Updates the set with the union of itself and others

References

- https://realpython.com/what-can-i-do-with-python/
- https://www.w3schools.com/python/python_getstarted.asp
- https://www.programiz.com/python-programming/first-program
- https://www.geeksforgeeks.org/
- https://www.w3resource.com/python-exercises/list/

Thank you for listening!