Tuples and Sets

can code communities

Data Structures

A list is a type of data structure.

In Python, and programming in general, there are many different types of data structures. Typically, data structures are collections, and also iterables - you can iterate through each item of them. They are a way to store multiple values in one place.

There are many different types of data structures. In Python, the main collections are lists, tuples, sets, and dictionaries.

Tuples

A tuple is a collection that is ordered and unchangeable.

It's similar to a list, but you can't add or remove items, and round brackets are used instead of square brackets: ( )

Tuples are faster than lists, and used to store constants - data that will not be changed. If you know you won't be changing the value of a collection, use a tuple.

Tuples can also be used as dictionary keys, or used to return multiple values from a function.

Using Tuples

Tuple Methods:

- `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.

You can index tuples with square brackets to access items, just like a list.

```
my_tuple = (54, 12)    ->    my_tuple[0] is 54
```

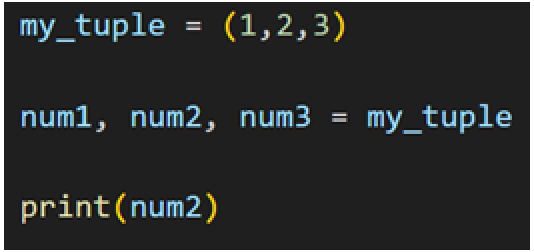
You can loop through tuples using a for loop.

You can add two tuples together using the + operator.

```
tuple1 = ("a", "b", "c")
tuple2 = ("d", "e")            ->    tuple3 is ("a", "b", "c", "d", "e")
tuple3 = tuple1 + tuple2
```

Tuple Unpacking

A tuple can be unpacked into multiple variables. You must have a variable for each item in the tuple.



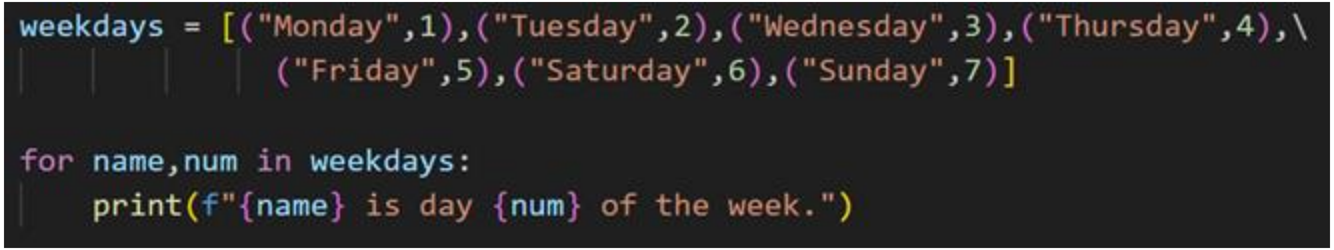
```python
my_tuple = (1,2,3)

num1, num2, num3 = my_tuple

print(num2)
```

What is the output of this code?

The main way this is used is in for loops. If you have a list of tuples, you can go through the elements in each of the tuples efficiently using tuple unpacking.



```python
weekdays = [("Monday",1),("Tuesday",2),("Wednesday",3),("Thursday",4),\
              ("Friday",5),("Saturday",6),("Sunday",7)]

for name,num in weekdays:
    print(f"{name} is day {num} of the week.")
```

What is the output of this code?

Exercise - Tuples

You have a tuple of numbers:

```
numbers = (1,2,3,4,5,6,7,8,9,10,11,12)
```

You have a tuple of months:

```
months = ("January","February","March","April","May","June",
"July","August","September","October","November","December")
```

Use these tuples to make a list of tuples where each tuple contains a number and the month it corresponds to, like this: `[("January",1),...,("December",12)]`

Now print each month and its number using tuple unpacking in a for loop. The first line of output should look like this:

```
January is month 1 of the year.
```

Set

A set is a collection that is unordered, unchangeable*, and unindexed.

* The items are unchangeable, but you can add and remove items.

Sets do not allow duplicates, so they are used to store a set of unique values. You use curly brackets for sets: { }

Because sets are unordered, you can't index them like a list. They don't have indexes at all. You can still loop through a set with a for loop.

Set Methods

`add()` Add an item to a set, like appending to a list

`clear()` Remove all the items in a set

`copy()` Returns a copy of a set

`difference()` Get the difference between two sets

`intersection()` Get the intersection between two sets (A set of items that appear in both)

`isdisjoint()` Returns whether two sets have an intersection or not

`remove()` Removes the specified element from the set

`symmetric_difference()` Get the symmetric difference between two sets

`union()` Get the union of two sets (A set of items that appear in either or both)

https://www.w3schools.com/python/python_ref_set.asp

Set Operations

Sets have some methods similar to lists like add() and remove(), but they also have some special methods specific to sets.

Union, intersection, difference, and symmetric difference are all logical operations that can be applied to sets. They are part of a field of math called set theory.

Each of these operations is applied to two or more sets.

Let's say `setA = {0,1,2,3}` and `setB = {2,3,4,5}`

- **Union:** The union of set A and set B is all the items that appear in **either** set A **or** set B. Can also be represented by the `|` operator.

  `setA.union(setB)` is the same as writing `setA | setB`.

  The result is a new set: `{0,1,2,3,4,5}`

Set Operations, continued

Let's say `setA = {0,1,2,3}` and `setB = {2,3,4,5}`

- **Intersection:** All the items that appear in **both** set A **and** set B. Shortcut: `&`

  `setA.intersection(setB)` is the same as writing `setA & setB`.

  The result is a new set: `{2,3}`

- **Difference:** The items that appear in set A, but **not** set B. Shortcut: –

  `setA.difference(setB)` is the same as writing `setA - setB`

  The result is a new set: `{0,1}`

- **Symmetric difference:** The items that appear in set A or set B, but **not both**. Can be thought of as the opposite of intersection. Shortcut: `^`

  `setA.symmetric_difference(setB)` is the same as writing `setA ^ setB`

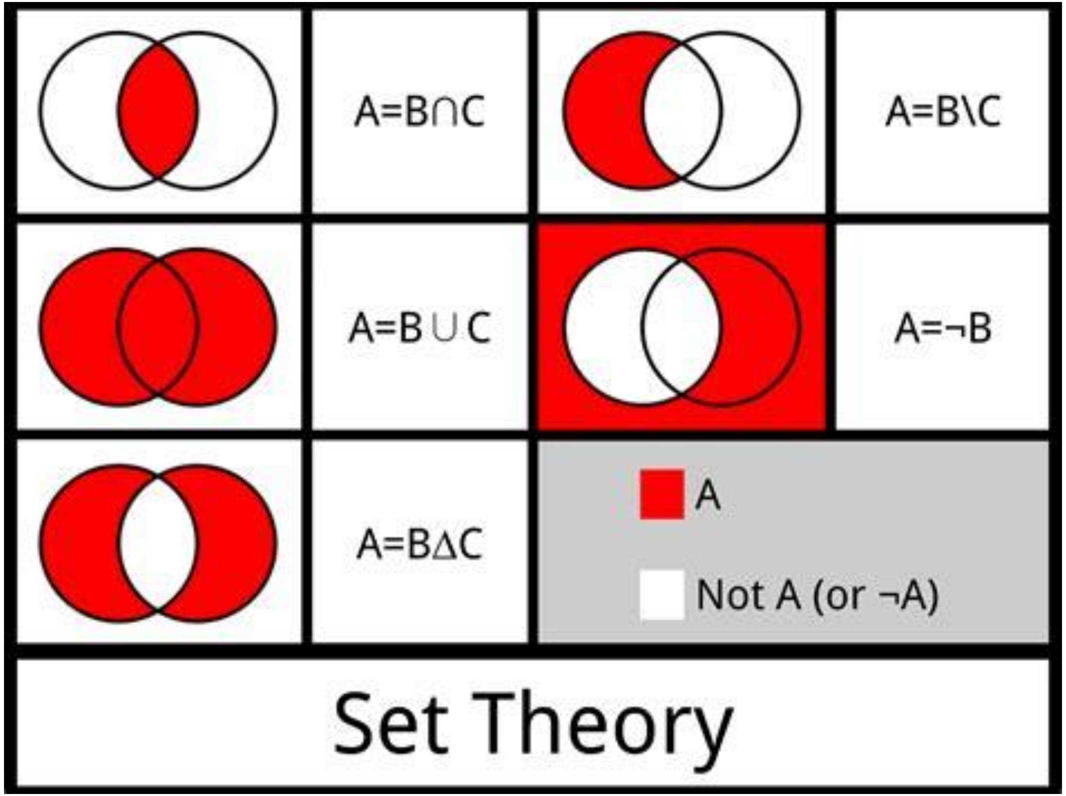
  The result is a new set: `{0,1,4,5}`

Set Operations in Set Theory

Intersection (B&C)

Union (B|C)

Symmetric Difference (B^C)



Difference (B-C)

Negation - Not possible with Python sets

Exercise - Sets

You work for a sales company and must generate a **set** of all customers who get a certain discount. The criteria for getting a discount is that they're over 60 years old and have made at least 5 purchases.

You have a **set** of customers over 60, and a **set** of customers who have made at least 5 purchases. Use a **set operation** to output a set of customers that fit both criteria for the discount. You can do this in one line of code.

Example:

```
over_60_years = {'Dominic', 'Linda', 'Simone', 'Swathi', 'Olaf'}
over_5_purchases = {'Finn', 'Simone', 'Aaron', 'Dominic'}
```

Output: `{'Dominic', 'Simone'}`

Exercise - Sets

You work at a company where some people know Python, some people know JavaScript, and some people know both.

In a loop, prompt the user to input an employee name, whether they know Python, and whether they know JavaScript. Use this to build two sets: a set of employees that know Python, and a set of employees that know JavaScript.

Use set operators to compute the following sets:

- The set of employees that know both Python and JavaScript
- The set of employees that know JavaScript, but not Python
- The set of employees that know Python or JavaScript, but not both

Lists vs. Tuples vs. Sets

Type	Brackets	Order	Immutable/ Mutable	Allow Duplicates?
List	[ ]	Ordered	Mutable	Yes
Tuple	( )	Ordered	Immutable	Yes
Set	{ }	Unordered	Immutable items, but you can add/remove items	No

Discussion: When would you choose to use a list, tuple, or set?

Resources

Tuples: https://www.w3schools.com/python/python_tuples.asp

Sets: https://www.w3schools.com/python/python_sets.asp

Shortcut operators for sets: https://www.geeksforgeeks.org/python-set-operations-union-intersection-difference-symmetric-difference/#