CIS 412 L Artificial Intelligence Lab
Topic - 03
Introduction to Python for Machine Learning

+88 01831-661534

jehadfeni@gmail.com

Python Tuples

Python Tuple is used to store the sequence of immutable Python objects. Tuple is similar to the python list. The main difference is list can be changed but a tuple cannot be change after creation. That means tuple is immutable.

A tuple is a collection which is ordered and **unchangeable**.

Creating a tuple

A tuple can be written as the collection of comma-separated (,) values enclosed with the small () brackets. The parentheses are optional but it is good practice to use. A tuple can be defined as follows.

Access tuple

- Tuple items can be access by index number inside brackets .
- Tuple items can be accessed by negative indexing
- Tuple items can be accessed by range of indexes
- Checking if an item is exist or not exist using membership operators

Modifying a tuple

- Since tuple is immutable we cannot modify a tuple directly but means we cannot add, update or remove elements of a tuple. If we need to modify a tuple at any cost we can follow a trick.
- The trick is you need to convert the tuple into a python list and perform necessary modifications on the list and then again convert the modified list into tuple.

Unpacking tuple items

- During creating a tuple we assign values to the tuple this process is called packing. But python also support the extraction of the tuple values into variables this is known as unpacking.

Looping and joining the tuples

Looping:

Besides, accessing by index number we can access a tuple items through loop.
We can use for loop, while loop and enhanced for loop to do that.

Joining tuples:

Two or more tuples can be joined by using a + operator

Multiplying a tuple items:

If you want to multiply the content of a tuple a given number of times, you can use the *. operator

Basic tuple operations

Operator	Description	Example
Repetition	The repetition operator enables the tuple elements to be repeated multiple times.	`T1*2 = (1, 2, 3, 4, 5, 1, 2, 3, 4, 5)`
Concatenation	It concatenates the tuple mentioned on either side of the operator.	`T1+T2 = (1, 2, 3, 4, 5, 6, 7, 8, 9)`
Membership	It returns true if a particular item exists in the tuple otherwise false	`print (2 in T1) prints True.`

Iteration	The for loop is used to iterate over the tuple elements.	`for i in T1:`
		`  print(i)`
		Output
		`1`
		`2`
		`3`
		`4`
		`5`

Length	It is used to get the length of the tuple.	`len(T1) = 5`

Tuple built in functions

SN	Function	Description
1	cmp(tuple1, tuple2)	It compares two tuples and returns true if tuple1 is greater than tuple2 otherwise false.
2	len(tuple)	It calculates the length of the tuple.
3	max(tuple)	It returns the maximum element of the tuple
4	min(tuple)	It returns the minimum element of the tuple.
5	tuple(seq)	It converts the specified sequence to the tuple.

Where to use tuple?

- Using tuple instead of list is used in the following scenario.
- 1. Using tuple instead of list gives us a clear idea that tuple data is constant and must not be changed.
- 2. Tuple can simulate a dictionary without keys. Consider the following nested structure, which can be used as a dictionary.

List vs Tuple

SN	List	Tuple
1	The literal syntax of list is shown by the [].	The literal syntax of the tuple is shown by the ().
2	The List is mutable.	The tuple is immutable.
3	The List has the a variable length.	The tuple has the fixed length.
4	The list provides more functionality than a tuple.	The tuple provides less functionality than the list.
5	The list is used in the scenario in which we need to store the simple collections with no constraints where the value of the items can be changed.	The tuple is used in the cases where we need to store the read-only collections i.e., the value of the items cannot be changed. It can be used as the key inside the dictionary.
6	The lists are less memory efficient than a tuple.	The tuples are more memory efficient because of its immutability.

Python set

- A Python set is the collection of the unordered items. Each element in the set must be unique, immutable, and the sets remove the duplicate elements. Sets are mutable which means we can modify it after its creation.
- Unlike other collections in Python, there is no index attached to the elements of the set.
- We can create a set by using curly braces ({}) or using set() method.

Accessing set elements

- A set element cannot be accessed by using an index or a key but we can use loop to access the elements.

Manipulating Set

We cannot update the value of a set item after creation but we can add new items to the set.

Add item:

We can use add() method to add single item to a set and can use update() method to add more than one item or a collection to a set.

Remove item:

We can use remove() or discard() to remove a particular item from the set. On the other hand we can use pop() method to remove the last element of the set.

Clear set contents:

We can use clear() method to clear the set elements

Deleting the set:

We can use del keyword to delete the entire set.

Loop and join sets

- **Loop:**

Since set is unordered we can loop through a set only using enhanced for loop.

- **Join**

We can use union() method to obtaining a new set containing elements of the both sets.

We can use update() method to insert the all items from one set to another set.

Keep only duplicate contents of two sets:

The intersection_update() method will keep only the items that are present in both sets.

Keep all but not duplicate contents of two sets:

The symmetric_difference_update() method will keep only the elements that are NOT present in both sets.

Python built-in set methods

SN	Method	Description
1	add(item)	It adds an item to the set. It has no effect if the item is already present in the set.
2	clear()	It deletes all the items from the set.
3	copy()	It returns a shallow copy of the set.
4	difference_update(....)	It modifies this set by removing all the items that are also present in the specified sets.
5	discard(item)	It removes the specified item from the set.
6	intersection()	It returns a new set that contains only the common elements of both the sets. (all the sets if more than two are specified).
7	intersection_update(....)	It removes the items from the original set that are not present in both the sets (all the sets if more than one are specified).
8	Isdisjoint(....)	Return True if two sets have a null intersection.
9	Issubset(....)	Report whether another set contains this set.
10	Issuperset(....)	Report whether this set contains another set.
11	pop()	Remove and return an arbitrary set element that is the last element of the set. Raises KeyError if the set is empty.
12	remove(item)	Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError.
13	symmetric_difference(....)	Remove an element from a set; it must be a member. If the element is not a member, raise a KeyError.
14	symmetric_difference_update(....)	Update a set with the symmetric difference of itself and another.
15	union(....)	Return the union of sets as a new set. (i.e. all elements that are in either set.)
16	update()	Update a set with the union of itself and others.

Dictionary

- Dictionaries are used to store data values in key: value pairs. A dictionary is a collection which is ordered*, changeable and does not allow duplicates.
- We can say that a dictionary is the collection of key-value pairs where the value can be any Python object.
- We can create dictionary by using multiple key-value pairs enclosed with the curly brackets {}, and each key is separated from its value by the colon (:)
- Or we can use bult in dict() method to create dictionary.

Access dictionary items

- We can access the items of a dictionary by referring to its key name, inside square brackets.
- Another way is to use get() method to access an element of a dictionary.
- If you don’t know the keys of a dictionary you can use keys() method that will return a list of all keys in a dictionary.

Manipulating the dictionary

- **Add element:**

We can add new item to the dictionary is done by using a new index key and assigning a value to it. Or we can use update() method.

- **Update element**

We can change the value of a specific item by referring to its key name or using update() method.

- **Remove elements**

We can use pop() method to remove a specific key from the dictionary or we can use popitem() method to remove last inserted item from the dictionary.

We can use The del keyword removes the item with the specified key name or we can use the same keyword to delete the entire dictionary.

We can use clear() method to clear the elements of a dictionary or making the dictionary empty.

Iterating the dictionary

We can iterate a dictionary using for loop in various way-

- To access/retrieve all the keys of the dictionary
- To access/retrieve all the keys of the dictionary using keys() method
- To access /retrieve all the values of the dictionary
- To access /retrieve all the values of the dictionary using values() method
- To access/retrieve both keys and pairs or the whole items using items() method

Dictionary methods

SN	Function	Description
1	cmp(dict1, dict2)	It compares the items of both the dictionary and returns true if the first dictionary values are greater than the second dictionary, otherwise it returns false.
2	len(dict)	It is used to calculate the length of the dictionary.
3	str(dict)	It converts the dictionary into the printable string representation.
4	type(variable)	It is used to print the type of the passed variable.

Dictionary functions cont...

SN	Method	Description
1	dic.clear()	It is used to delete all the items of the dictionary.
2	dict.copy()	It returns a shallow copy of the dictionary.
3	dict.fromkeys(iterable, value = None, /)	Create a new dictionary from the iterable with the values equal to value.
4	dict.get(key, default = "None")	It is used to get the value specified for the passed key.
5	dict.has_key(key)	It returns true if the dictionary contains the specified key.
6	dict.items()	It returns all the key-value pairs as a tuple.
7	dict.keys()	It returns all the keys of the dictionary.
8	dict.setdefault(key,default= "None")	It is used to set the key to the default value if the key is not specified in the dictionary
9	dict.update(dict2)	It updates the dictionary by adding the key-value pair of dict2 to this dictionary.
10	dict.values()	It returns all the values of the dictionary.
11	len()	
12	popItem()	
13	pop()	
14	count()	
15	index()	