Data structure Lab

Tuples

Dr. Halah H. Mahmoud
Computer science department / College of Science for Women

Tuples

- Python Tuples are like a list. It can hold a sequence of items. The difference is that it is immutable
-
- **How to Create a Python Tuple?**
- To declare a Python tuple, you must type a list of items separated by commas, inside parentheses. Then assign it to a variable.
- >>> percentages=(90,95,89)
- You should use a tuple when you don't want to change just an item in future.

Tuples

1. Python Tuples Packing

- You can also create a Python tuple without parentheses. This is called tuple packing.
- >>> b= 1, 2.0, 'three'

2. Python Tuples Unpacking

- Python tuple unpacking is when you assign values from a tuple to a sequence of variables in python.
- >>> percentages=(99,95,90,89,93,96)
- >>> a,b,c,d,e,f=percentages
- >>> c

- **Output:** 90

Tuples

3. Creating a tuple with a single item

- Until now, we have seen how easy it is to declare a Python tuple. But when you do so with just one element, it may create some problems. Let's take a look at it.
- >>> a=(1)
- >>> type(a)

- **Output**
- <class 'int'>

Tuples

- Wasn't the type() method supposed to return class 'tuple'?
- To get around this, we add a comma after the item.
- >>> a=(1,)
- >>> type(a)

- **Output**
- <class 'tuple'>

Also, like a list, a Python tuple may contain items of different types.

- >>> a=(1,2.0,'three')

Tuples

- **How to Access Python Tuple?**

1. Accessing the entire tuple

- To access a tuple in python, just type its name.
- >>> percentages

Output: (90, 95, 89)

- Or, pass it to the print statement.
- >>> print(percentages)
- **Output:** (90, 95, 89)

2. Accessing a single item

- To get a single item from a Python tuple, use its index in square brackets. Indexing begins at 0.
- >>> percentages[1]
- **Output:** 95

Tuples

- **Slicing a Tuple in Python**
- If you want a part(slice) of a tuple in Python, use the slicing operator [].
- >>> percentages=(99,95,90,89,93,96)
- 1. Positive Indices
- When using positive indices, we traverse the list from the left.
- >>> percentages[2:4]

- **Output:** (90, 89)
- This prints out items from index 2 to index 3 (4-1) (items third to fourth).
- >>> percentages[:4]

Tuples

- **Slicing a Tuple in Python**
- **Output:** (99, 95, 90, 89)

- This prints out items from the beginning to the item at index 3.
- >>> percentages[4:]

- **Output:** (93, 96)
- This prints out items from index 4 to the end of the list.
- >>> percentages[2:2]

- **Output:** ()
- However, this returns an empty Python tuple.

Tuples

2. Negative indexing

- Now, let's look at negative indexing. Unlike positive indexing, it begins traversing from the right.
- >>> percentages[:-2]

- **Output:** (99, 95, 90, 89)
- This prints out the items from the tuple's beginning to two items from the end.
- >>> percentages[-2:]
- **Output:** (93, 96)

Tuples

2. Negative indexing

- This prints out items from two items from the end to the end.
- >>> percentages[2:-2]

- **Output:** (90, 89)
- This prints out items from index 2 to two items from the end.
- >>> percentages[-2:2]:

- **Output:** ()

Tuples

2. Negative indexing

- This last piece of code, however, returns an empty tuple. This is because the
- start(-2) is behind the end(2) in this case.
- Lastly, when you provide no indices, it prints the whole Python tuple.
- >>> percentages[:]
- **Output:** (99, 95, 90, 89, 93, 96)

Tuples

- **Deleting a Python Tuple**
- a Python tuple is immutable. This also means that you can't delete just a part of it. You must delete an entire tuple, if you may.
- >>> del percentages[4]

- **Output**
- Traceback (most recent call last):File "<pyshell#19>", line 1, in <module>
- del percentages[4]
- TypeError: 'tuple' object doesn't support item deletion

Tuples

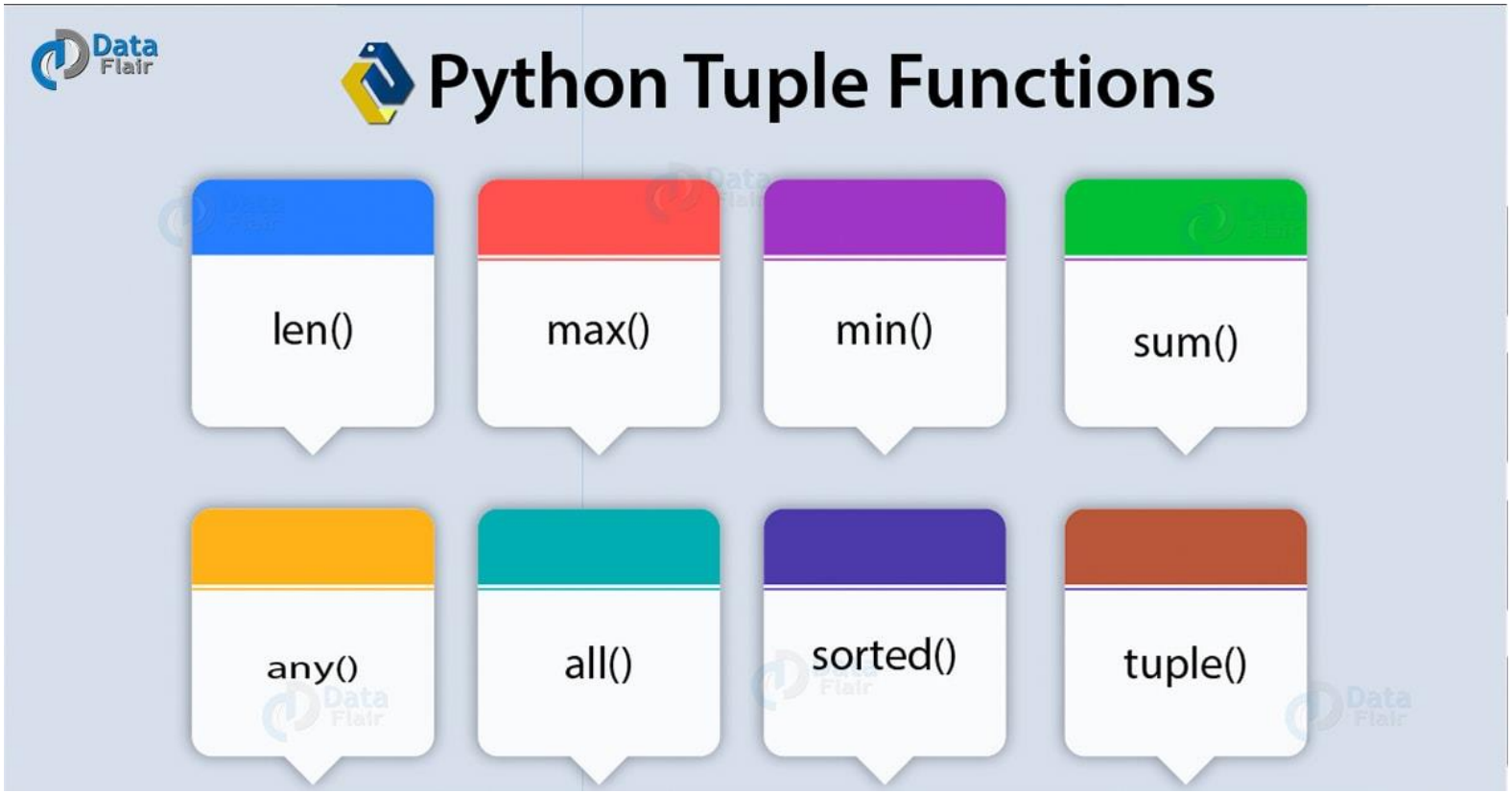
- **Deleting a Python Tuple**
- So, deleting a single element didn't work. Let's try deleting a slice.
- >>> del percentages[2:4]
- **Output**
- Traceback (most recent call last):File "<pyshell#20>", line 1, in <module>del percentages[2:4]
- TypeError: 'tuple' object does not support item deletion

- deleting the entire tuple.
- >>> del percentages
- >>> percentages
- **Output**
- Traceback (most recent call last):File "<pyshell#40>", line 1, in <module>
- percentages
- NameError: name 'percentages' is not defined

Tuples

- **Reassigning Tuples in Python**
- As we discussed, a Python tuple is immutable. So let's try changing a value. But before that, let's take a new tuple with a list as an item in it.
- >>> my_tuple=(1,2,3,[4,5])
- Now, let's try changing the list [4,5]. Its index is 3.
- >>> my_tuple[3]=6
- **Output**
- Traceback (most recent call last):File "<pyshell#43>", line 1, in <module>
- my_tuple[3]=6
- TypeError: 'tuple' object does not support item assignment

Python Tuple Functions



Tuples

1. len()

- Like a list, a Python tuple is of a certain length. The len() function returns its length.
- >>> my_tuple
- **Output**
- (1, 2, 3, [6, 5])
- >>> len(my_tuple)
- **Output**
- 4
- It returned 4, not 5, because the list counts as 1.

Tuples

2. max()

- It returns the item from the tuple with the highest value.
- We can't apply this function on the tuple my_tuple, because ints cannot be compared to a list. So let's take yet another tuple in Python.
- >>> a=(3,1,2,5,4,6)
- >>> max(a)
- **Output**
- 6

- ax(('Hi','hi','Hello'))
- **Output**
- 'hi'

Tuples

3. min()

- Like the max() function, the min() returns the item with the lowest values.
- >>> min(a)
- **Output**
- 1
- As you can see, 1 is the smallest item in this Python tuple.

Tuples

- **4. sum()**
- This function returns the arithmetic sum of all the items in the tuple.
- >>> sum(a)
- **Output:** 21
- However, you can’t apply this function on a tuple with strings.
- >>> sum(('1','2','3'))

- **Output**
- Traceback (most recent call last):File “<pyshell#57>”, line 1, in <module>
- sum(‘1′,’2′,’3’))
- TypeError: unsupported operand type(s) for +: ‘int’ and ‘str’

Tuples

- **5. any()**
- If even one item in the tuple has a Boolean value of True, then this function returns True. Otherwise, it returns False.
- >>> any(('','0',''))

- **Output:** True
- The string ‘0’ does have a Boolean value of True. If it was rather the integer 0, it would’ve returned False.
- >>> any(('',0,''))

- **Output:** False

Tuples

6. all()

- Unlike any(), all() returns True only if all items have a Boolean value of True. Otherwise, it returns False.
- >>> all(('1',1,**True**,''))
- **Output:** False

Tuples

7. sorted()

- This function returns a sorted version of the tuple. The sorting is in ascending order, and it doesn't modify the original tuple in Python.
- >>> sorted(a)
- **Output:** [1, 2, 3, 4, 5, 6]

Tuples

8. tuple()

- This function converts another construct into a Python tuple. Let's look at some of those.
- >>> list1=[1,2,3]
- >>> tuple(list1)
- **Output:** (1, 2, 3)
- >>> string1="string"
- >>> tuple(string1)
- **Output:** ('s', 't', 'r', 'i', 'n', 'g')
- How well would it work with sets?
- >>> set1={2,1,3}
- >>> tuple(set1)
- **Output:** (1, 2, 3)
- >>> set1
- **Output:** {1, 2, 3}

Tuples

- **Python Tuple Methods**
- A method is a sequence of instructions to perform on something. Unlike a function, it does modify the construct on which it is called. You call a method using the dot **operator in python.** Let's learn about the two in-built methods of Python.
- **1. index()**
- This method takes one argument and returns the index of the first appearance of an item in a tuple. Let's take a new tuple.
- >>> a=(1,2,3,2,4,5,2)
- >>> a.index(2)
- **Output**
- 1
- As you can see, we have 2s at indices 1, 3, and 6. But it returns only the first index.

Tuples

- Python Tuple Methods
- **2. count()**
- This method takes one argument and returns the number of times an item appears in the tuple.
- >>> a.count(2)
- **Output**
- 3

Tuples

- **Python Tuple Operations**

1. Membership

- We can apply the ‘in’ and ‘not in’ operators on items. This tells us whether they belong to the tuple.
- >>> 'a' **in** tuple("string")
- **Output:** False
- >>> 'x' not **in** tuple("string")
- **Output:** True

2. Concatenation

- Like we’ve previously discussed on several occasions, concatenation is the act of joining. We can join two tuples using the concatenation operator ‘+’.
- >>> (1,2,3)+(4,5,6)
- **Output:** (1, 2, 3, 4, 5, 6)
- Other arithmetic operations do not apply on a tuple.

Tuples

- **Python Tuple Operations**

3. Logical

- All the logical operators (like >,>=,..) can be applied on a tuple.
- >>> (1,2,3)>(4,5,6)
- **Output:** False
- >>> (1,2)==('1','2')
- **Output:** False
- As is obvious, the ints 1 and aren't equal to the strings '1' and '2'. Likewise, it returns False.

4. Identity

- Remember the 'is' and 'is not' operators we discussed about in our tutorial on Python Operators? Let's try that on tuples.
- >>> a=(1,2)
- >>> (1,2) is a
- **Output:** False