# Lists and Tuples

## Exercises

### Week 6

Prior to attempting these exercises ensure you have read the lecture notes and/or viewed the video, and followed the practical. You may wish to use the Python interpreter in interactive mode to help work out the solutions to some of the questions.

Download and store this document within your own filespace, so the contents can be edited. You will be able to refer to it during the test in Week 7.

Enter your answers directly into the highlighted boxes.

For more information about the module delivery, assessment and feedback please refer to the module within the MyBeckett portal.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## ©2021 Mark Dixon / Tony Jenkins

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Would you describe the following Python statement as a **function call**? Or a **method call**?

names.reverse()

*Answer:*

The statement names.reverse() in Python is a method call.

In Python, a method is a function that belongs to an object, and it is accessed using dot notation (.).

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a Python statement that appends a single element to the end of the specified *List* using a **method** call.

prices = [2.65, 7.65, 8.25, 9.56]

*Answer:*

prices = [2.65, 7.65, 8.25, 9.56]

prices.append(12.34)

Write another statement that appends three elements to the end of the specified *List* using a single **method** call.

*Answer:*

prices = [2.65, 7.65, 8.25, 9.56]

prices.extend([12.34, 15.67, 18.90])

Now write a for loop that *iterates* over each value in the list and prints it to the screen.

*Answer:*

prices = [2.65, 7.65, 8.25, 9.56, 12.34, 15.67, 18.90]

for price in prices:

print(price)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is a method that changes the contents of the associated value referred to as a **mutator**? Or an **accessor**?

*Answer:*

A method that changes the contents of the associated value is typically referred to as a mutator method.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What would the contents of the primes list look like after execution of the following statements?

primes = [ 2, 3, 5, 7, 11, 13, 17, 19 ]

primes.pop()

*Answer:*

[2, 3, 5, 7, 11, 13, 17]

The pop() method in Python removes and returns the last element from a list.

primes.reverse()

*Answer:*

[19, 17, 13, 11, 7, 5, 3, 2]

After executing primes.reverse(), the order of elements in the primes list will be reversed.

primes.remove(7)

*Answer:*

[19, 17, 13, 11, 5, 3, 2]

After executing primes.remove(7), the method will find and remove the value 7 from the list.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Provide an example of how the insert() method could be used to add a value of 10 to the beginning of the list shown below.

temps = [ 32, 46, 95, 10, 50 ]

*Answer:*

temps = [32, 46, 95, 10, 50]

# Adding the value 10 to the beginning of the list

temps.insert(0, 10)

Now write a statement that uses an *accessor* method to find the index of the value 95 within the list.

*Answer:*

temps = [10, 32, 46, 95, 10, 50]

# Finding the index of the value 95 in the list

index\_of\_95 = temps.index(95)

print("Index of value 95 in the list:", index\_of\_95)

Finally write a statement that uses another *accessor* method to count how many times the number 10 appears within the list.

*Answer:*

temps = [10, 32, 46, 95, 10, 50]

# Counting the occurrences of the value 10 in the list

count\_of\_10 = temps.count(10)

print("Number of occurrences of value 10 in the list:", count\_of\_10)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What would be stored in the list samples after the following statements were executed?

samples = [ 100.2, 100.6, 99.2, 765.2, 900.2, 400 ]

samples = samples.reverse()

*Answer:*

samples = [400, 900.2, 765.2, 99.2, 100.6, 100.2]

Explain why this is the case.

*Answer:*

After executing samples.reverse(), the samples list would be reversed in place

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a Python program that uses a **List-Comprehension** to produce the same list as the following code -

values = []

for n in range(100,200):

values.append(x\*x)

*Answer:*

values = [n \* n for n in range(100, 200)]

Now, amend your code so that it only includes even numbers.

*Answer:*

values = [n\*n for n in range(100, 200) if n % 2 == 0]

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

What is the *data-type* of the following value?

info = ("Ken", "bae-192", 62)

*Answer:*

The value assigned to the variable info appears to be a tuple in Python.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Is a Tuple **mutable** or **immutable**?

*Answer:*

In Python, tuples are immutable data structures. Once a tuple is created, its elements cannot be changed, added, or removed. This immutability distinguishes tuples from lists, which are mutable and allow modifications after creation.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a statement that creates a Tuple that contains a single element.

*Answer:*

single\_element\_tuple = (42,)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

Write a single Python statement that **unpacks** the following Tuple into three variables, called x, y and z.

coord = (100, 200, 150)

*Answer:*

x, y, z = coord

Write another statement that uses indexing to access the second element of the Tuple and store it in a variable called ‘height’

*Answer:*

coord = (100, 200, 150)

height = coord[1] # Accessing the second element (index 1) and storing it in 'height'

Finally write a ‘for’ loop that prints each value within the Tuple.

*Answer:*

coord = (100, 200, 150)

# Using a for loop to print each value in the tuple

for value in coord:

print(value)

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When a Tuple (or any sequence) type value is being passed as an argument to a function, what single character can be used as a prefix to force the sequence to be **unpacked** prior to the call being made?

*Answer:*

In Python, when a tuple (or any sequence) needs to be unpacked and its elements passed as arguments to a function call, you can use the asterisk (\*) operator as a prefix to force the sequence to be unpacked.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

When discussing Tuples the phrase **heterogeneous** is sometimes used to describe the type of stored values. What does this mean in practice?

*Answer:*

In the context of tuples, the term "heterogeneous" refers to the ability of tuples to store elements of different data types within the same tuple.

What sister phrase is often used to refer to the type of values stored within a List? And what does this mean?

*Answer:*

The sister phrase often used to refer to the type of values stored within a list is "homogeneous." In contrast to tuples being heterogeneous, lists are considered homogeneous in the context of the types of values they store.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

## **Exercises are complete**

Save this logbook with your answers. Then ask your tutor to check your responses to each question.