# 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.

---



---

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.

---

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.append(5.75)

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

*Answer:*

Multiple elements can be added to the end of a list in a single method call by using the extend() function.

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, 1.23, 5.67, 3.45]
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:*

The 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:*

The contents of the primes list will look like [2, 3, 5, 7, 11, 13, 17].

```
primes.reverse()
```

*Answer:*

The contents of the primes list will look like[17, 13, 11, 7, 5, 3, 2].

```
primes.remove(7)
```

*Answer:*

The contents of the primes list will look like [17, 13, 11, 5, 3, 2].

---

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, 50]
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, 50]
index_of_95 = temps.index(95)

print(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, 50]
count_of_10 = temps.count(10)

 occurrences of 10 in the list
print(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:*

After it is executed, the value kept in the samples variable would be None.

Explain why this is the case.

*Answer:*

The reverse() function returns None and makes changes to the list in place. Samples become None when the result is assigned to them, not the inverted list.

---

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)]
print(values)
```

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]
print(values)
```

---

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

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

*Answer:*

The *data-type* of the following value is **Tuples**.

---

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

*Answer:*

Tuple is **immutable.**

---

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

*Answer:*

```
my_single_element_tuple = (7,)
```

---

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 = (100, 200, 150)
```

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

*Answer:*

```
values = (100, 200, 150)
height = values[1]
```

Finally write a '`for`' loop that prints each value within the Tuple.

*Answer:*

```
values = (100, 200, 150)
for value in values:
 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:*

Python functions can be called with an asterisk (*) character as a prefix to cause the sequence to be unpacked when they receive a tuple or any other sequence as an argument. The "unpacking" operator is the term for this.

---

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

*Answer:*

When discussing Tuples, the term "heterogeneous" refers to the ability of tuples to store elements of different data types.

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." It means that all elements in the list are of the same data type.

---

# _ Exercises are complete

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