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 following Python statement 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:

The python statement that appends a single line element to the end of the specified list using a method call is: prices.append(10.30)

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

Answer:

The statement that appends three elements to the end of the specified list using a single method call is: prices.extend([11.45, 12.30, 13.15])

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

Answer:

For loop that iterate over each value in the list and prints it to the screen is given below:

```
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:

Method that changes the contents of the associated value referred to as a mutator.

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 pop() method removes the last element of the list. So, the output is:
[2, 3, 5, 7, 11, 13, 17]

```
primes.reverse()
```

Answer:

The reverse() method reverses the order of the list elements. So, the output is:
[17, 13, 11, 7, 5, 3, 2]

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

Answer:

The value 7 is removed from the list.
[2, 3, 5, 11, 13, 17, 19]

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:

The insert() helps to add a value at a specific index in the list. So:
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.index(95)

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

Answer:

```
temps.count(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 the execution of reverse() method, the following statement will be stored as:
[400, 900.2, 765.2, 99.2, 100.6, 100.2]

Explain why this is the case.

Answer:

Because, when the reverse() method is done it reverses the order of the elements in place. Meaning first will move to the last position, second to second last position , third to third last position and so on till the entire list is reversed.

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 = [x * x for x in range(100, 200)]
```

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

Answer:

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

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

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

Answer:

The data type of the following value is Tuple.

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

Answer:

A Tuple is immutable.

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

Answer:

```
single_element_tuple = (12,)
```

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
```

where,
x=100. y=200, z=150

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

Answer:

```
height = coord[1]
```

This will store the second element of the tuple coord in variable height.

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

Answer:

```
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:

The asterisk(*) is used as a prefix to unpack a sequence like a tuple before passing its elements as separate arguments to a function.

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

Answer:

In practice, heterogeneous means a tuple can store elements of different data types, such as integers, strings, floats, and even other tuples or lists.

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

Answer:

Homogenous sister phrase is often used to refer to the type of values stored within the list. The sister phrase "homogeneous" refers to a list typically containing elements of the same data type, though mixed types are also allowed.

Exercises are complete

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