CMM201_Week3_lab_solved

October 4, 2019

1 Week 3 Laboratory

In this Jupyter notebook you will have the opportunity to practice with the Python data types and data structures. Moreover, you can keep familiarising yourself with the Jupyter notebook environment.

1.1 Data types and data structures

We will start with the variable *uni* which contains a string

```
In [1]: uni = "Robert Gordon University"
        print(uni)
```

```
Robert Gordon University
```

```
In [2]: # 1. Use this cell to print the length of the string, as well as the first and last ch
        # Hint 1: a string works like a tuple/list of letters
        # Hint 2: Negative numbers indicate the index of elements from right to left (i.e. in
        print('Length of the string uni is: ',len(uni))
        print('First character of uni is: ', uni[0])
        print('Last character of uni is: ',uni[-1])
```

```
Length of the string uni is:  24
First character of uni is:  R
Last character of uni is:  y
```

In Python you can access a "range" of characters by using the *:* symbol. For instance, to access the first two characters in *uni* string, you can use the following command:

```
In [3]: uni[0:2] # characters from position 0 (included) to 2 (excluded)
```

```
Out[3]: 'Ro'
```

```
In [4]: # Use this cell to access the 2nd, 3rd and 4th characters
        uni[2:5]
```

```
Out[4]: 'ber'
```

```
In [5]: #Use this cell to access the first five characters
        uni[:5]
```

```
Out[5]: 'Rober'
```

```
In [6]: # Use this cell to access from the third to the last character
        uni[2:]
```

```
Out[6]: 'bert Gordon University'
```

```
In [7]: #Use this cell to access the last three characters
        uni[-2:]
```

```
Out[7]: 'ty'
```

Just as with variables containing numbers, we can do operations with variables containing strings. For instance, we can add a string to another.

```
In [8]: uni + ' rocks!'
```

```
Out[8]: 'Robert Gordon University rocks!'
```

Which other mathematical operations can you do with strings? Try in the following cell

```
In [9]: # Use this cell to try to use any other mathematical operation with strings
        # Hint: you can do a calculation using a string and an int
        uni * 2
```

```
Out[9]: 'Robert Gordon UniversityRobert Gordon University'
```

Now you will work with a list called *squares*, which contains squared numbers from 1 to 5

```
In [10]: squares = [1, 4, 9, 16, 25]
         print(squares, type(squares))
```

```
[1, 4, 9, 16, 25] <class 'list'>
```

You will also work with a list called *things* which contains elements of different types (it even contains the *squares* list

```
In [11]: things = [1, uni, 9.86, False, squares]
         things
```

```
Out[11]: [1, 'Robert Gordon University', 9.86, False, [1, 4, 9, 16, 25]]
```

```
In [12]: # Use this cell to replace the third elements from the squares list into a 9.001
         squares[2]=9.001
         print(squares)
```

```
[1, 4, 9.001, 16, 25]
```

```
In [13]: # use this cell to replace the fifth element of the things list into the squares list
         things = [1, uni, 9.86, False, squares]
         things
```

```
Out[13]: [1, 'Robert Gordon University', 9.86, False, [1, 4, 9.001, 16, 25]]
```

```
In [14]: # Use this cell to add two more squares (36 and 49) to the squares list
         squares = squares + [36, 49]
         squares
```

```
Out[14]: [1, 4, 9.001, 16, 25, 36, 49]
```

```
In [15]: # Use this cell to append the list ['a','b'] to the things list
         things = things + [['a','b']]
         things.append(['a','b'])
         print(things)
```

```
[1, 'Robert Gordon University', 9.86, False, [1, 4, 9.001, 16, 25], ['a', 'b'], ['a', 'b']]
```

Just as with numbers, we can compare lists

```
In [16]: # Compare two lists
         [1,2,3] == [1,3,3]
```

```
Out[16]: False
```

```
In [18]: # Use this cell to check if the element 4 is in the squares list
         # Hint: You can use the in operator to quickly check
         4 in squares
```

```
Out[18]: True
```

```
In [19]: # Use this cell to delete the third element from the squares list
         # Hint: You can use the del operator
         del squares[2]
         squares
```

```
Out[19]: [1, 4, 16, 25, 36, 49]
```

As we saw before, a tuple cannot be modified (is immutable), however the values contained can be unpacked into other variables.

```
In [20]: tupthings = (1, uni, 9.86, False, squares)
         var1,var2,var3,var4,var5 = tupthings
         print(var1,var2,var3,var4,var5)
```

```
1 Robert Gordon University 9.86 False [1, 4, 16, 25, 36, 49]
```

1.2 Other data structures

1.2.1 Ranges

Ranges are "lists" of numbers which can be defined with the *range()* function.

```
In [22]: r = range(10)
         type(r)
```

```
Out[22]: range
```

You can convert a range into a list.

```
In [23]: r = list(r)
         r
```

```
Out[23]: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
```

You can also define ranges between numbers using two inputs: the initial and the final value.

```
In [25]: # Use this cell to define a range between 5 (inluded) and 10 (excluded).
         # Transform the range into a list to verify the result.
         list(range(5,10))
```

```
[5, 6, 7, 8, 9]
```

```
In [26]: # Use this cell to define a range between 5 (inluded) and 10 (excluded) with a step o
         # Hint, the range function accepts the step as a third input.
         # Transform the range into a list to verify the result.
         list(range(5,10,2))
```

```
Out[26]: [5, 7, 9]
```

```
In [28]: # Use this cell to define a range between 9 (included) and 1 (excluded) in descending
         # Transform the range into a list to verify the result.
         list(range(9,1,-1))
```

```
Out[28]: [9, 8, 7, 6, 5, 4, 3, 2]
```

1.2.2 Sets

Sets are unordered collection of elements. These are defined using curly brackets {}

```
In [29]: basket = {'apple', 'orange', 'pear', 'orange', 'banana'}
         print(basket)
```

```
{'orange', 'banana', 'pear', 'apple'}
```

Notice that even if 'orange' was defined twice, it was only included once in the set.

```
In [30]: # Use this cell to check if there are grapes in the basket
         'grape' in basket
```

```
Out[30]: False
```

1.2.3 Dictionaries

Like lists, but indexed with a key.

```
In [31]: dictStudent = {'Name': 'Alex', 'Age': 27, 'Grade': 'A'}
         dictStudent

Out[31]: {'Age': 27, 'Grade': 'A', 'Name': 'Alex'}

In [32]: len(dictStudent)

Out[32]: 3
```

Instead of the index, you can get an element by using the key.

```
In [33]: # Requesting a specific element from the dictionary by using the key
         print(dictStudent['Name'])

Alex
```

1.3 Typing

You may recall from last week's activity that Python has particular rules when it comes to typing numbers into variables. Now that we know more data types and data structures, let's see some other typing rules.

```
In [34]: # Example with integers
         x = 3
         y = x
         print('Value of x is', x)
         print('Value of y is', y)

Value of x is 3
Value of y is 3


In [35]: # Since integers are IMMUTABLE, a new object '2' is created. Python removes the refer
         y=y-1
         print('Value of x is', x)
         print('Value of y is', y)

Value of x is 3
Value of y is 2


In [36]: # Example with lists
         x = [1,2]
         y = x
         print('Value of x is', x)
         print('Value of y is', y)
```

```
Value of x is [1, 2]
Value of y is [1, 2]
```

```
In [37]: # Since lists are MUTABLE, the command y[0]=0 changed the list as an object, not x!
         y[0]=0
         print('Value of x is', x)
         print('Value of y is', y)
```

```
Value of x is [0, 2]
Value of y is [0, 2]
```

```
In [38]: # Solution: copy
         x = [1,2]
         y = x.copy()
         y[0]=0
         print('Value of x is', x)
         print('Value of y is', y)
```

```
Value of x is [1, 2]
Value of y is [0, 2]
```