1 What is a boolean value? (Where are they used? What expressions generate them?)

A boolean value is a True/False, Yes/No, Heads/Tails, Success/Failure, or One/Zero value often used as a flag for flow control in a programming script. If-else blocks are key flow control statements that generate boolean values. For examples, if (True) do {this} else do {that}. Empty, none, zero, or non-existent elements generate False boolean values while populated elements and non-zero values (e.g. one) generate True boolean values.

2 Describe all of the parts of a for loop.

A for loop in Python consists of:

- A for statement with an element name and a list
- A code block to execute for each element in the list

Here is an example of a for loop syntax:

```
for letter in alphabet:
    code block     # notice this block must be indented
```

If alphabet contains all the letters from a to z then the code block will execute 26 times (one time for each of the letters). For each execution the letter from the list will be stored in the variable 'letter' in this example. It could be any name, but here I chose 'letter' to make the statement read well.

3 Some collections are mutable, others are not. Which collections are not mutable and why does it matter?

Python's collections are:

- tuples - an ordered immutable series of objects
- strings - an ordered immutable series of characters
- lists - an ordered mutable series of objects
- sets - mutable unique series of objects
- dicts - mutable unordered key-value pairs

It matters that tuples and strings are immutable so you have some guarantee that the ordered values will not get changed. It also matters because if you should try to alter elements of a tuple or string the compiler will complain and the edit will fail. Choosing the best data structure is up to you the programmer to get right.

4 What are two different *kinds* of strings in python and how is character escaping handled in each kind?

1. Normal strings which are defined by single or double quotes. Character escaping is done with a backslash which gives special meaning to the next character. For example, in a normal string \n will not be printed as '\n' but rather will start a new line.

2. Multiline strings which are defined by three quote marks. Character escaping for newline and tab work like normal with \n and \t. Also new lines are allowed. Lastly, with multiline strings we can use an combination of quotes without escaping them. For example, this works with multiline strings:

```
x = '''Marcus Aurelis said "A man's worth is no greater than his ambitions."'''
```

and print(x) generates:

Marcus Aurelis said "A man's worth is no greater than his ambitions."

as expected.

5 What is the *self* keyword? How is it used?

In Python *self* is the instantiation of the class being used. It is used to refer to the that object (i.e. the class instantiation). In object oriented programming objects have properties that can be accessed with dot notation. An example of this object property syntax in a class definition is:

self.name = 'Monty'

6 What is the __init__ function? When is it called?

__init__ is a class function that is run as soon as the class is instantiated. It is used to do any setup on the class employed. For example,

```
class CS519:
def __init__(self, student):
     self.student = student
def greetStudent(self):
     print('Meet the student ', self.student)

s = CS519('Monty')
s.greetStudent()
```

will print 'Meet the student Monty'
Notice that __init__ was never specifically called but rather automatically executed when s was instantiated.

7 If something is O(n log(n)) what does that mean?

Big O notation denotes time complexity for a process to run. O(n log(n)) means that for every element n, the run time will be modeled by the equation n log(n).

8 Why are base cases important in recursion?

A base case in recursion is any event for which the answer is already known. Armed with this information, the result for any base case does not have to be recalculated but rather can be looked up. This is especially important for large, complex algorithms and is the basis for dynamic programming.

1. You are trying to explain mutable vs immutable types to a colleague. You realize that an example would really help clear things up. Write a short program (5-10 lines would be fine) that demonstrates a major difference between mutable an immutable types. Include a paragraph describing what it demonstrates.

In Python tuples are ordered immutable sequences. Their ordered mutable cousin are lists. This example the attempts to change both the tuple and the list then prints the results. Defining both a tuple and a list with the variables a, b, and c works fine. Printing out mutableList and immutableTuple show each contain order, indexable elements as expected. When you try to change an element of the list to a string that works fine and the results can be printed but, when you attempt the same thing with the tuple you get an error.

```
a = 1
b = 2
c = 3

mutableList = [a, b, c]
print('mutableList is: ', mutableList)

immutableTuple = (a, b, c)
print('immutableTuple is: ', immutableTuple)

mutableList[1] = 'rabbit'
print(mutableList)

immutableTuple[1] = 'rabbit'  # Results in compiler error here
print(immutableTuple)
```

2. Your friend is bragging about how their computer is so fast it can compute anything! Prove them wrong. Write a function that is O(2^n) or slower. In an accompanying paragraph explain why your function is that slow.

The recursive example given in module 9 is a great example of an O(2^n) function. Given a modest number of approximately 40 will take quite a while on my machine. Increasing that number to 108, for example, will take a ridiculous amount of time and/or computing power. The reason it is so slow is has to calculate the answer for every permutation from the bottom up every time. The resulting tree of permutations is enormous so will take a long time.

```
def fib(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fib(n-1) + fib(n-2)

fibonacci = fib(108)
print(fibonacci)
```