

Assignment 1 – Part 2: Python

Question 1

Code a function called `first_div_16`

- ACCEPT two positive integers, $n1$ and $n2$, as inputs
- RETURN the first number in `range(n1, n2)` that is divisible by 16.
- However, if no number in the range is divisible by 16, RETURN 0.

Code:

```
import math

def first_div_16(n1, n2):
    """A function that finds, and returns, the first number in between
    the range of n1 and n2 that is divisible by 16"""

    firstDivisibleNumber = math.ceil(n1/16) * 16 #find the first number >= n1 that is divisible by 16
    if firstDivisibleNumber > n2: #check that the first divisible number is within the range
        return 0
    else:
        return firstDivisibleNumber

def printResultsFor(rangeStart,rangeEnd):
    print("range start: ", rangeStart, " , range end:", rangeEnd,
          ", result:", first_div_16(rangeStart, rangeEnd))

rangeStart = 1
rangeEnd = 78
printResultsFor(rangeStart,rangeEnd)

rangeStart = 5
rangeEnd = 25
printResultsFor(rangeStart,rangeEnd)

rangeStart = 33
rangeEnd = 48
printResultsFor(rangeStart,rangeEnd)

rangeStart = 32
rangeEnd = 50
printResultsFor(rangeStart,rangeEnd)

rangeStart = 49
rangeEnd = 50
printResultsFor(rangeStart,rangeEnd)
```

Description:

For this function, I calculate the first number greater than or equal to $n1$ that is divisible by 16. Then I compare this with $n2$ to check if this value is within the specified range. If so, this value is returned, otherwise, 0 is returned.

To calculate the first number divisible by $16 \geq n1$, I make use of the python math module. Firstly, I divide $n1$ by 16, to get a result of how many times that 16 goes into $n1$. Secondly, I use the `math.ceil`

function to round up to the next whole number. Multiplying this whole number by 16 I get the first number divisible by 16 that is equal to or greater than n1.

I chose this implementation to be inclusive of the n1 and n2 range. So, if n1 was 16 then the result would be 16, and if n1 was 28 and n2 was 32, the result would be 32. I was not sure however if this was specified in the question, so I thought this was the most appropriate implementation.

The code for this function is found in the file named FirstDivInRange.py

Results:

```
Last login: Mon Nov 23 17:37:40 2020 from 192.168.10.156
codio@prince-bruce:~/workspace$ python3 FirstDivInRange.py
range start: 1 , range end: 78 , result: 16
range start: 5 , range end: 25 , result: 16
range start: 33 , range end: 48 , result: 48
range start: 32 , range end: 50 , result: 32
range start: 49 , range end: 50 , result: 0
codio@prince-bruce:~/workspace$
```

Question 1 Iterative Alternative Approach

Code:

```
def first_div_16(n1,n2):
    """A function that finds, and returns, the first number in between
    the range of n1 and n2 that is divisible by 16"""

    for num in range(n1,n2 +1):#This range is inclusive of the start and end number
        if num % 16 == 0: #The number is divisible by 16 if the remainder of the division is 0
            return num

    return 0 #This will only be called if a division by 16 is not possible

rangeStart = 16
rangeEnd = 50
print("range start: ", str(rangeStart), " , range end:", str(rangeEnd),
      ", result:" , str(first_div_16(rangeStart,rangeEnd)) )
```

Description:

For this alternative approach, I iterate over all the numbers within the range provided, checking if they are divisible by 16. If one of the numbers we check is divisible by 16, then we return that number. If not, we return 0. The modulo operator is used to check whether a number is divisible by 16. The modulo operator returns the remainder of a division. If the remainder is 0 we know the number is divisible by 16. This code is inclusive of the start and end range values, so if they are a divisible by 16, they will be returned also.

The code for this function is found in the file named FirstDivInRangeAlt.py

Results:

```
codio@prince-bruce:~/workspace$ python3 FirstDivInRange.py
range start: 17 , range end: 31 , result: 0
codio@prince-bruce:~/workspace$ python3 FirstDivInRange.py
range start: 16 , range end: 31 , result: 16
codio@prince-bruce:~/workspace$ python3 FirstDivInRange.py
range start: 17 , range end: 32 , result: 32
codio@prince-bruce:~/workspace$ python3 FirstDivInRange.py
range start: 1 , range end: 20 , result: 16
codio@prince-bruce:~/workspace$
```

Question 2

Code a function called `halve_to_2`

- ACCEPT one numeric input.
- If the number ≤ 0 , RETURN -1.
- If the number > 0 , divide that integer over-and-over by 2 until it becomes smaller than 2.
- RETURN that smaller-than-2 number, e.g. input of 4 will yield 1 (4->2->1), 5 yields 1.25 (5->2.5->1.25) etc.

Code:

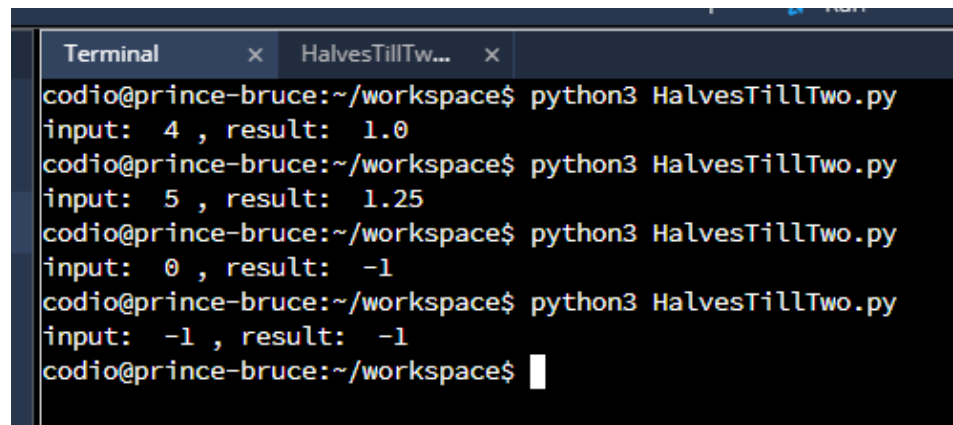
```
def halve_to_2(num):  
    """This function will halve the input value until it is  
    smaller than 2, the remaining number will be returned"""  
  
    if num <= 0:  
        return -1  
  
    while(num >= 2): #Keep dividing by 2 until the number is less than 2  
        num = num / 2  
  
    return num  
  
testNumber = -3  
print("input: ", testNumber, ", result: ", str(halve_to_2(testNumber)))
```

Description:

This function makes use of the while operator to divide the input by 2 until the value is less than 2.

The code for this function is found in the file named: `HalvesTillTwo.py`

Results:



```
Terminal x HalvesTillTw... x  
codio@prince-bruce:~/workspace$ python3 HalvesTillTwo.py  
input: 4 , result: 1.0  
codio@prince-bruce:~/workspace$ python3 HalvesTillTwo.py  
input: 5 , result: 1.25  
codio@prince-bruce:~/workspace$ python3 HalvesTillTwo.py  
input: 0 , result: -1  
codio@prince-bruce:~/workspace$ python3 HalvesTillTwo.py  
input: -1 , result: -1  
codio@prince-bruce:~/workspace$
```

Question 3

Code a function called `string_expansion`.

- ACCEPT a non-empty string as input
- RETURN a string that contains every other character, $2n+2$ times, where n is the original index of the letter. e.g. Input of "Hello" should result in "HHlllllllooooooooooooo". Input of "ROBErt" should result in "RRBBBBBBrrrrrrrrrrr".

Code:

```
def string_expansion(string):  
    """This function returns a string where every other  
    charecter is repeated  $2n+2$  times, where  $n$   
    is the index of the charecter in the original string"""  
  
    res = ""  
    for index in range(len(string)):  
        if index % 2 == 0: #index % 2 == 0 is true for every odd number  
            res += string[index] * ((2 * index) + 2)  
    return res  
  
testInput = "Hello"  
print("input: ", testInput, ", result: ", string_expansion(testInput))
```

Description:

In this function I create a result string that I initialize as empty before I add to it in the for loop. Then in the for loop, I iterate through every character of the string. If the index of the current character is odd, I skip the character, if the index is even, then I append $2n + 2$ instances of the character (where n is the index of the char in the original string). I make use of python's string multiplication to implement the formula in code.

The code for this function is found in the file named `StringExpansion.py`

Results:

```
codio@prince-bruce:~/workspace$ python3 StringExpansion.py  
input: ROBErt , result: RRBBBBBBrrrrrrrrrr  
codio@prince-bruce:~/workspace$ python3 StringExpansion.py  
input: Hello , result: HHlllllllooooooooooooo  
codio@prince-bruce:~/workspace$
```

Question 3 Iterative Alternative Approach

An alternative to using python's string multiplication, would be the use of a for loop control structure to construct the multiplied string as shown in this example:

```

def string_expansion(string):
    """This function returns a string where every other
    charecter is repeated 2n+2 times, where n
    is the index of the charecter in the original string"""

    res = ""
    for index in range(len(string)):
        if index % 2 == 0: #index % 2 == 0 is true for every odd number

            newStr = ""
            for i in range(2 * index):
                newStr += string[index]
            newStr += string[index] + string[index]

            res += newStr

    return res

testInput = "Hello"
print("input: ", testInput, ", result: ", string_expansion(testInput))

```

Although I prefer the use of python string multiplication, this approach may be easier to understand for programmers not familiar with python's string multiplication feature.

This code can be found in [StringExpansionAlt.py](#)

Question 4

Code a function called `item_count_from_index`.

- ACCEPT two inputs, a list and an integer-index
- RETURN a count (number) of how many times the item at that index appears in the list.
- However, if the integer-index is out of bounds for the list RETURN the empty string ("") (e.g. list of 3 items, index of 5 is out of bounds)

Code:

```
def item_count_from_index(list, index):
    """This function counts the number of occurrences of the item at the
    provided index in the list"""

    if index > len(list) - 1 or index < 0: #if the index is out of bounds, return an empty string
        return ""

    itemToCount = list[index] #the index is a 0 base indexed

    count = 0
    for item in list:
        if item == itemToCount: #count the number of times the item occurs in the list
            count += 1
    return count

testInput = ['b', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'a', 'b']
testIndex = 0
print("input list:", testInput, ", input index:", testIndex,
      ", result:", item_count_from_index(testInput, testIndex))
```

Description:

First this code checks if the index provided, as the second argument, is outside the bounds of the list. Note that I check the list length -1, this is because the list length is not zero indexed, while the argument is.

Next store the item to search for in the `itemToCount` variable. Then I iterate through all the values in the list, for each item, I check whether the value equals the `itemToCount` value, if so, I add 1 to the count.

After getting the item count using the for loop, I return the count value.

Note: the index function argument is 0 indexed.

The code for this function is found in the file named `CountOccurence.py`

Results:

```
codio@prince-bruce:~/workspace$ python3 CountOccurence.py
input list: ['a', 'b', 'a', 2, 'a', '2', 'b', 'b', 'a', []] , input index: 1 , result: 3
codio@prince-bruce:~/workspace$ python3 CountOccurence.py
input list: [0, 1, 2, 3, 4, 5, 6, 7, 2, 9, 10] , input index: 2 , result: 2
codio@prince-bruce:~/workspace$
```

Question 5

Code a function called `length_times_largest`.

- ACCEPT a list as input
- RETURN the length of the list times the largest integer (not float) in the list.
- However, if the list does not contain an integer, RETURN the empty string (`""`).

Code:

```
def length_times_largest(list):
    """A function that returns the length of the list times by
    the largest integer in the list"""

    largest = 0
    containsInt = False
    setFirstLargestInt = False #this enables handling negative integers

    for item in list:
        if type(item) == int:#we are only looking for integers
            containsInt = True
            if not setFirstLargestInt:#if this is the first integer found, it is the new largest
                largest = item
            elif item > largest:
                """if this is not the first integer found,
                only set a new value if this is larger than the previous largest integer"""
                largest = item

    if not containsInt:#if the this list not contain any integers, return an empty string
        return ""

    return len(list) * largest

listInput = ["a","agsaga",324.421421,-10,10,'']
print("inputList:", listInput, " , result: ", length_times_largest(listInput))
```

Description:

In this function I first iterate through all the items in the list to find the largest integer. While iterating though, I check if each item is an integer, then check if it is larger than the previous largest integer. If so, I set the largest variable to the new value. For the first integer found however I skip the comparison with the current largest value as that value is not yet set, instead taking the first value as the starting point for all future comparisons. I make use of a Boolean that tracks whether the list contains any integers. When iterating through the list, if an integer is found, then the Boolean is set to True, otherwise the value remains False. This Boolean is used to check whether to return an empty string as per the requirements. I often start with a value of zero for the largest value starting point, but in this case that would exclude negative values, so I used the Boolean logic to enable a negative first “largest” value.

The code for this function is found in the file named `ListTimesLargest.py`

Results:


```
Terminal X ListTimesLarge...
codio@prince-bruce:~/workspace$ python3 ListTimesLargest.py
inputList: ['a', 'agsaga', 324.421421, -10, 10, ''] , result: 60
codio@prince-bruce:~/workspace$ python3 ListTimesLargest.py
inputList: ['a', 'agsaga', 324.421421, -10, -10, ''] , result: -60
codio@prince-bruce:~/workspace$ python3 ListTimesLargest.py
inputList: ['a', 'agsaga', 324.421421, '', '', ''] , result:
codio@prince-bruce:~/workspace$ python3 ListTimesLargest.py
inputList: ['a', 'agsaga', 324.421421, 1, '', ''] , result: 6
codio@prince-bruce:~/workspace$
```