CompSci 101 Assignment 3

Due: 4:30pm, May 21

Worth: 3% of your final mark

Topic: lists

This assignment is marked out of 30

CodeRunner3 Assignments

Assignment 3 is completely marked on **CodeRunner3**. **There is no other submission required.**

https://www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/

Assignment 3 – Complete 8 functions

For Assignment 3, I have posted programs containing the skeletons and testing code for the 8 assignment questions. Download these programs from the CompSci 101 assignments website:

https://www.cs.auckland.ac.nz/courses/compsci101s1c/assignments/

Develop the solution to each function in each program.

Once you are happy that a function executes correctly, submit **the whole function** to **CodeRunner3**. You will receive immediate feedback from CodeRunner3 telling you if you have passed the tests for that question. You can submit as many times as you like. You need to submit one function at a time.

CompSci 101 Assignment 3

Some helpful information about Questions 1, 2 and 3

A3 Q1 – get_numbers()

parameters – a list of integers

returns – a new list

Function creates and returns a **new list** containing all the **odd** numbers from the parameter list which are **NOT** exactly divisible by 3.

```
print("1.", get_numbers([23, 3, 6, 5, 12, 9, 7, 4]))
print("2.", get_numbers([87, 77, 49, 21, 4, 80, 51]))
print("3.", get_numbers([9, 30, 27, 36]))
print("4.", get_numbers([]))
```

```
1. [23, 5, 7]
2. [77, 49]
3. []
4. []
```


A3 Q1 – get_numbers()

How to create a new list, how to add elements to the new list.

1. Create an empty Python list:

```
result_list = []
```

2. Fill the list with the relevant elements using the `append()` method. Usually this happens over and over, i.e. inside a loop:

```
. . .
    result_list.append(an_element)
```

3. Usually the filled Python list is returned as a result of the function:

```
return result_list
```


A3 Q1 – get_numbers()

How to test if a number is exactly divisible by another number.

1. Check if there is zero remainder when a number is divided by another number, use the % operator

```
number1 = . . . #some integer
number2 = . . . #some integer

if number1 % number2 == 0:
    #do what is needed if number1 is exactly
    #divisible by number2
```


A3 Q1 – get_numbers()

Use the `append()` method to add elements to the end of the list.

Add a single element to the end of the list using the `append()` method:

```
result_list = []
an_element = …
result_list.append(an_element)
```

A3 Q2 - get_funny_average ()

parameter - a list of numbers

returns – the average (rounded to the nearest whole number) of the list elements excluding the smallest number, the largest number and zeroes. Returns 0 if no numbers to average.

```
print("1. [3, 12, 0, 25, 1]: ",
                    get_funny_average([3, 12, 0, 25, 1]))
print("2. [-6, -32, 2, 0, -51, 1, 0, 0]: ",
              get_funny_average([-6, -32, 2, 0, -51, 1, 0, 0]))
print("3. [56, 0, 2, 0, 22]: ",
              get_funny_average([56, 0, 2, 0, 22]))
print("4. [5, 0, 2, 0]: ", get_funny_average([5, 0, 2, 0]))
```

```
1. [3, 2, 0, 25, 1]:  8
2. [-6, -32, 2, 0, -51, 1, 0, 0]:  -12
3. [56, 0, 2, 0, 22]:  22
4. [5, 0, 2, 0]:  0
```


A3 Q2 - get_funny_average ()
How to get the smallest, biggest number in a list?

The `min()` the `max()` functions can be used to get the minimum and maximum elements in a Python list:

```
numbers_list = [3, 2, 0, 25, 1]
smallest = min(numbers_list)
biggest = max(numbers_list)
```


A3 Q2 - get_funny_average ()
How to get the total of all the numbers in a list?

The `sum()` function can be used to get the total of all the elements in a Python list:

```
numbers_list = [3, 2, 0, 25, 1]
total = sum(numbers_list)
```


A3 Q2 - get_funny_average ()
… excluding zeroes.

Hint: create a new Python list and only add the non-zero elements.

```
numbers_list = [3, 2, 0, 25, 0, 0, 1]
non_zero_list = []

   #Fill non_zero_list with all non zero
   #elements from the numbers_list list
```

A3 Q2 - get_funny_average ()
When to return zero.

Once the list with all the non-zero elements has been created, the smallest and largest elements can be removed. BUT what if the list is empty? What if the list has only one element? What if the list has only two elements?

4 Examples `[5, 2]` or `[5]` or `[5, 2, 7]` or `[]`

```
def some_function(…, …):
    . . .
    if condition1:
        return 0 #stop executing the
                 #function and return 0

    #continue the function code knowing that
    #condition1 is not true
```


A3 Q3 - get_fail_pass_average()

parameters – a list of integers

returns – a Python list made up of two elements:

the average of all the marks which are less than 50, followed by the average of all the marks which are 50 or more (both averages are rounded to the nearest whole number). Average is -1 if there are no marks in the category.

```
print("1.", get_fail_pass_average([63, 65, 33]))
print("2.", get_fail_pass_average([63, 62, 100, 100]))
print("3.", get_fail_pass_average([33, 42, 20, 10]))
print("4.", get_fail_pass_average([]))
```

```
1. [33, 64]
2. [-1, 81]
3. [26, -1]
4. [-1, -1]
```


A3 Q3 - get_fail_pass_average()

Hint: create two lists and add relevant elements to each list.

```
list_below = []
list_pass = []


    #Either add the element to list_below
    #or to list_pass
```


A3 Q3 - get_fail_pass_average()

Average is -1 if there are no marks in the category.

Work out the averages for the two lists. BUT what if either of the lists is empty? Division by 0 will generate an error!

For Example, `[63, 62, 100, 100]`

```
list_below = []
list_pass = []

. . .

#Work out averages BUT check first in
#case the average needs to be set to -1
```

A3 Q3 - `get_fail_pass_average()`

How to get the total of all the numbers in a list? … the number of elements in the list?

The `sum()` function can be used to get the total of all the elements in a Python list. The `len()` function can be used to get the total of all the elements in a Python list.

```
numbers_list = [3, 2, 0, 25, 1]

total = sum(numbers_list)

number_of_elements = len(numbers_list)
```


A3 Q3 - `get_fail_pass_average()`

How to create a Python list made up of two elements:

A Python list can be initialised using square brackets and the initial elements, each separated by commas, e.g.

```
element1 = . . .
element2 = . . .
element3 = . . .

result_list = [element1, element2, element3]
```