# quiz-list

September 16, 2024

## 0.1 QUIZ - List

**Q 1:**

Define a function named **sum_of_list**.

The parameter will be a list.

The function will add all the elements in the list and return the summation result.

[1]:
```
# Q 1:

# ---- your solution here ----
def sum_of_list(lis):
    total = 0
    for i in lis:
        if type(i) == int:
            total += i
    return total

# call the function you defined
nums = [1, 2, 3, 4, 5, 'a']
summation = sum_of_list(nums)
print(summation)
```

```
15
```

---

**Q 2:**

Define a function named **two_levels_sum**.

The parameter will be a list of two levels (nested list).

The function will add all the elements including the nested lists and return the summation result.

**Hints:** * isdigit() * type()

[2]:
```
# Q 2:

# ---- your solution here ----
def two_levels_sum(lis):
    total = 0
    for i in lis:
        if type(i) == int:
            total += i
        elif type(i) == list:
            for j in i:
                total +=j
    return total

# call the function you defined
a_list = [[5, 8], [1, 4, 7], [10], 3, 'a']
summation = two_levels_sum(a_list)
print(summation)
```

```
38
```

---

**Q 3:**

Define a function named **squares**.

It will calculate the square of each item in the list and append it to a new list.

It will return the new list of squares.

[4]:
```
# Q 3:

# ---- your solution here ----
from math import sqrt
def squares(lis):
    squared = []
    for i in lis:
        squared.append(sqrt(i))
    return squared

# call the function you defined
numbers = [1,2,3,4,5]
result = squares(numbers)
print(result)
```

```
[1.0, 1.4142135623730951, 1.7320508075688772, 2.0, 2.23606797749979]
```

---

**Q 4:**

Define a function named **sum_of_squares**.

It will calculate the squares of items first and will add them to create items of a new list.

Each item in the new list will be the sum of items from the old list starting from index 0 up to the item's index.

For example in the new list the item in index 4 is going to be the sum of squares of items at indices 0-1-2-3-4 of parameter list.

And it will return the new list.

**Hints:** * use sum_of_list function you defined in Q1

[7]:
```
# Q 4:

# ---- your solution here ----
from math import sqrt
def sum_of_squares(lis):
    total = 0
    for i in lis:
        total += sqrt(i)
    return total

# call the function you defined
a_list = [1,2,3,4,5]
result = sum_of_squares(a_list)
print(result)
```

```
8.382332347441762
```

---

**Q 5:**

Define a function that return the difference between items at odd indices (1,3,5…) and items at even indices (0,2,4…).

The function name will be **odd_even_difference**.

**Hints:** * index starts from zero * use lists and indices only * do not use loop * use sum_of_list function you defined in Q1

[9]:
```
# Q 5:

# ---- your solution here ----
def odd_even_difference(lis):
    odd_indices_sum = sum_of_list(lis[1::2])

    even_indices_sum = sum_of_list(lis[0::2])

    return odd_indices_sum - even_indices_sum

# call the function you defined
a_list = [1,2,3,4,5,6]
diff = odd_even_difference(a_list)
print(diff)
```

```
3
```

---

**Q 6:**

Define a function that crop the list parameter.

The function will be **crop_the_list**.

Cropping means remove the first and the last elements from the list.

The function will **mutate the original list parameter** namely it will modify the list in-place.

It will not return any value.

Test the function and prove it mutates the list which you pass as argument.

[11]:
```python
# Q 6:

# ---- your solution here ----
def crop_the_list(lis):
    if len(lis) > 1:
        lis.pop(0)
        lis.pop(-1)

# call the function you defined
nums = [1,2,3,4,5,6,7]
print('the list before passing as parameter:', nums)
crop_the_list(nums)
print('the list before passing as parameter:', nums)
```

```
the list before passing as parameter: [1, 2, 3, 4, 5, 6, 7]
the list before passing as parameter: [2, 3, 4, 5, 6]
```

---

**Q 7:**

Define a function named **sort_the_list**.

It will take a list and sort type (is_reverse) as parameters.

**is_reverse** will be boolean and it will decide if the sort will be ascending or descending. It's default value will be False. * if is_reverse is True -> sort will be descending * if is_reverse is False -> sort will be ascending

The function will return the new sorted list.

**Hints:** * sorted() * no loops * do not mutate the original list

[12]:
```python
# Q 7:

# ---- your solution here ----
def sort_the_list(lst, is_reverse=False):
    return sorted(lst, reverse=is_reverse)

```

```
# call the function you defined
a = [12, 4, 2, 1, 6, 3, 45]
print('a:', a)

sorted_a = sort_the_list(a, False)
print('sorted - ascending', sorted_a)

sorted_a = sort_the_list(a, True)
print('sorted - descending', sorted_a)

print('a after sort_the_list function call:', a)
```

```
a: [12, 4, 2, 1, 6, 3, 45]
sorted - ascending [1, 2, 3, 4, 6, 12, 45]
sorted - descending [45, 12, 6, 4, 3, 2, 1]
a after sort_the_list function call: [12, 4, 2, 1, 6, 3, 45]
```

---

**Q 8:**

The Consecutive Numbers in Math are actually Lists in Python.

**Gauss' Method for Summing Consecutive Numbers** is a method that gives you the summation of numbers.

And that's how Gauss' Method works: * First it sorts the list in ascending order * Then sorts the list in descending order * Adds the items in both lists at the same index * Divides the overall summation by 2 (because each element is written 2 times)

Example: Sum of all integers from 1 to 10.

In this question you are going to sum all the number starting from 2 up to 100 with step size 3.

You should use Gauss' Method in Python to calculate the overrall sum.

The function name will be **gauss** and the parameters and default values will be: * start -> 1 * end -> 100 * step_size -> 2

**Hints:** * use range() to create the list

[13]:
```
# Q 8:

# ---- your solution here ----
def gauss(start=1, end=100, step_size=2):
    numbers = list(range(start, end + 1, step_size))

    ascending = sorted(numbers)
    descending = sorted(numbers, reverse=True)

    pairwise_sum = sum([a + b for a, b in zip(ascending, descending)])
    return pairwise_sum / 2

# call the function you defined
conseq_sum = gauss(2, 100, 3)
print(conseq_sum)
```

```
1650.0
```

---