Lab 5

Four fundamental algorithms

知识点/Keys

1. Returns sum of all positive integers smaller than n

```
def summation(n):
    '''
    (number)->number
    '''
    odd_sum = 0
    for num in range(n):
        odd_sum = odd_sum + num
    return odd_sum
```

2. Returns a product of all positive integers smaller than n

```
def product(n):
    '''
    (number -> number)
    '''
    prod = 1
    for num in range(1,n):
        prod = prod * num
    return prod
```

3. Sum of numbers in a given list a. A solution with loop over elements

```
def sum_list_v1(a):
    '''
    (list)->num
    '''
    list_sum = 0
    for item in a:
        list_sum = list_sum + item
    return list_sum
```

4. Sum of numbers in a given list a. A solution with a loop over indices

```python
def sum_list_v2(a):
    '''
    (list)->num
    '''
    list_sum = 0
    for i in range(len(a)):
        list_sum = list_sum + a[i]
    return list_sum
```

Task 4 and Programming Exercise 1

1. After studying the previous four fundamental functions, how would you modify them to only sum odd numbers (or odd list elements)? Try yourself and then see next step for solutions.
2. Open the file called four_functions.py Copy/paste, one by one, Example 1 to 4 into Python visualizer. Run through each example and understand how the solutions work and how the variables change in the loops. As always, you can find python visualizer here (make sure you choose Python 3) http://www.pythontutor.com/visualize.html#mode=edit
3. Programming exercise: Write a function called ah(l,x,y) that given a list l, and integers x and y such that x <=y, returns two numbers. The first is the number of elements of l that are between x and y (including x and y). The second number is the minimum element of l that is between x and y (including x and y). Example test:

```
t=[5, 1, -2.5, 10, 13, 8]
ah(t, 2,11)
(3, 5)
```

1. 在学习了前面的四个基本函数之后，你会如何修改它们以仅对奇数（或奇数列表元素）求和？自己尝试一下，然后查看下一步的解决方案。
2. 打开名为 four_functions.py 的文件，将示例 1 到 4 逐一复制/粘贴到 Python 可视化工具中。运行每个示例并了解解决方案的工作原理以及变量在循环中的变化方式。与往常一样，你可以在这里找到 Python 可视化工具（确保选择 Python 3）http://www.pythontutor.com/visualize.html#mode=edit
3. 编程练习：编写一个名为 ah(l,x,y) 的函数，给定一个列表 l 以及整数 x 和 y，使得 x <=y，返回两个数字。第一个是 l 中介于 x 和 y 之间的元素个数（包括 x 和 y）。第二个数字是 l 中介于 x 和 y 之间的最小元素（包括 x 和 y）。

#方法一

```python
def ah(l,x,y):
    count = 0  # 用来计数符合条件的元素
    min_el = None  # 用来存储最小值

    # 使用for循环遍历列表中的每个元素
    for i in l:
        # 如果元素在x和y之间（包括x和y），则处理
```

```
        if x <= i <= y:
            count += 1  # 增加计数
            # 如果 min_element 是 None 或当前元素小于 min_element，更新最小值
            if min_el is None or i < min_el:
                min_el = i

    # 返回计数和最小值
    return (count, min_el)
```

方法二：

```
def ah(l,x,y):
    filt_num = [i for i in l if x <= i <= y] # 创建遍历集合l中的每个元素，并将它放到这个数组中
    count = len(filt_num) # 计算数组的长度
    min_num = min(filt_num) # 在数组中找最小数
    return(count, min_num)
```

Task 5

```
def test1():
    count = 0
    for i in range(-2, 14):
        print("*")
        print("**")
        count += 1
    print(count) #16,then times 3 = 48
    return
```

Task 6

```
def task2():
    fruits = ["apple","orange","banana","cherry"]
    for afruit in fruits:
        print(afruit, end = " ")

    for position in range(len(fruits)):
        print(fruits[position], end=" ")


    for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
        print("Hi ", name, "Please come to my party on Saturday !")
```

Task 7

```python
# task 7
def task3():
    for i in range(5,0,-1):
        num = 1
        for j in range(1,i+1):
            print(num, end="| ")
            num = num*2
        print()
```

Programming Exercise 2

A positive integer is called a perfect number if it is equal tothe sum of all of its positive divisors, excluding itself. Forexample, 6 is perfect number since 6=1+2+3. The next is28=1+2+4+7+12. There are four perfect numbers less than10,000. Write a program that prints all these four numbers.

Your program should have a function called is_perfect thattakes as input a positive integers and returns True if it is perfectand False otherwise.Once you are done. Modify your program so that it looks for allperfect numbers smaller than 35 million. What do you notice?

Assuming that you computer can do a billion instructions in a sec, canyou figure out how long, roughly, will it take your computer to find 5thperfect number (it is 33,550,336). Is the answer roughly: couple ofminutes, couple of hours, couple of days, .. weeks, months, years ?

What if you wanted to wait until it prints 6th perfect number, which is 8,589,869,056?

> 如果正整数等于其所有正因数（不包括自身）之和，则该正整数被称为完美数。例如，6 是完美数，因为 6=1+2+3。下一个是 28=1+2+4+7+12。有四个小于 10,000 的完美数。编写一个程序打印所有这四个数字。
>
> 您的程序应该有一个名为 is_perfect 的函数，该函数以正整数为输入，如果是完美数，则返回 True，否则返回 False。完成后。修改您的程序，使其查找小于 3500 万的所有完美数。你注意到了什么？
>
> 假设您的计算机可以在一秒钟内执行十亿条指令，您能算出您的计算机大约需要多长时间才能找到第五个完美数（它是 33,550,336）吗？答案大概是：几分钟、几个小时、几天……几周、几个月、几年？
>
> 如果你想等到它打印出第 6 个完全数字，即 8,589,869,056，该怎么办？

```python
# Programming Exercise 2
def is_perfect(n):
    sum_of_div = 1  # 1 是任何数的真因子
    for i in range(2, int(math.sqrt(n)) + 1):
        if n % i == 0:
            sum_of_div += i
            if i != n // i:  # 处理平方数的情况
                sum_of_div += n // i
    return sum_of_div == n
```

Programming Exercise 3a: Arithmetic progression

Recall that a sequence of numbers forms an arithmetic progression if the difference between every pair of consecutive numbers is the same. For example: -5, -1, 3, 7, 11forms an arithmetic progression since the difference between every pair of consecutive numbers is 4. On the contrary 5, 10, 15, 24, 29 is not an arithmetic progression since the difference between some consecutive pairs is 5 and some 4.
A sequence that has exactly one number is considered arithmetic, too. Write a function called arithmetic that takes as input a list of numbers and returns True if the numbers of the list form arithmetic progression. And False otherwise

> 回想一下，如果每对连续数字之间的差相同，则数字序列形成一个算术级数。例如：-5、-1、3、7、11 形成一个算术级数，因为每对连续数字之间的差为 4。相反，5、10、15、24、29 不是算术级数，因为一些连续对之间的差为 5，一些连续对之间的差为 4。
> 只有一个数字的序列也被认为是算术级数。编写一个名为 arithmetic 的函数，以数字列表作为输入，如果列表中的数字形成算术级数，则返回 True。否则返回 False

```
def arithmetic(l):
    '''l is a list of number'''
    if len(l) <= 2:
        return True

    diff = l[1] - l[0]

    for i in range(1,len(l) -1):
        if l[i + 1] - l[i] != diff:
            return False
    return True
```

Programming Exercise 3b: and now … is it sorted?

Now modify your method arithmetic slightly so that instead it tests if the numbers in the give lists are ordered for smallest to largest. Call the new function `is_sorted`

> 现在稍微修改一下方法算法，让它测试给定列表中的数字是否按从小到大的顺序排列。调用新函数 `is_sorted`

```
def is_sorted(n):
    if len(n) <= 1:
            return True

        # 遍历数组，检查每对相邻元素是否满足 arr[i] <= arr[i+1]
        for i in range(len(n) - 1):
            if n[i] > n[i + 1]:
                return False

        return True
```