Engineering Math 2

Laboratory Manual

LAB 4: Sequences and Series

Instruction:

1) Read through the lab manual, practice with the examples in the yellow boxes, and fill in the answer in ⬚.

2) After you finish, copy the page with the filled in-purple rectangles as pdf or image file.

3) Do all practice problems then save as the .ipynb file.

4) Submit all files from 2) and 3) to the Google classroom.

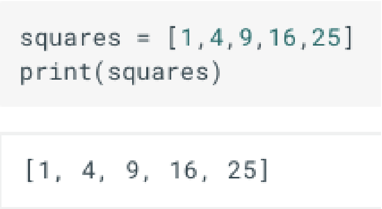
1. Sequences

The main sequence types in Python are lists, tuples and range objects. The main differences between these sequence objects are:

- Lists are mutable and their elements are usually *homogeneous* (things of the same type making a list of similar objects)
- Tuples are immutable and their elements are usually *heterogeneous* (things of different types making a tuple describing a single structure)
- Range objects are *efficient* sequences of integers (commonly used in for loops), use a small amount of memory and yield items only when needed

  - **Lists**

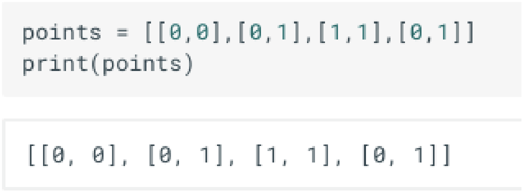
Create a list using square brackets [...] with items separated by commas. For example, create a list of square integers, assign it to a variable and use the built-in function print() to display the list:



```
squares = [1,4,9,16,25]
print(squares)
```

```
[1, 4, 9, 16, 25]
```

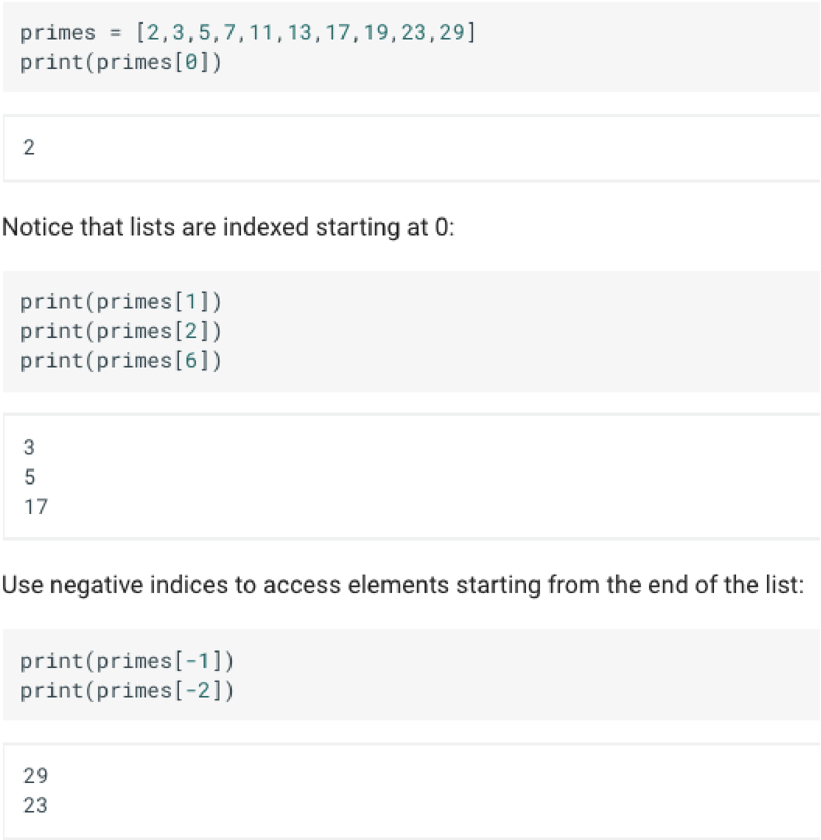
Lists may contain data of any type including other lists:



```
points = [[0,0],[0,1],[1,1],[0,1]]
print(points)
```

```
[[0, 0], [0, 1], [1, 1], [0, 1]]
```

Access the elements of a list by their index:



```
primes = [2,3,5,7,11,13,17,19,23,29]
print(primes[0])
```

```
2
```

Notice that lists are indexed starting at 0:

```
print(primes[1])
print(primes[2])
print(primes[6])
```

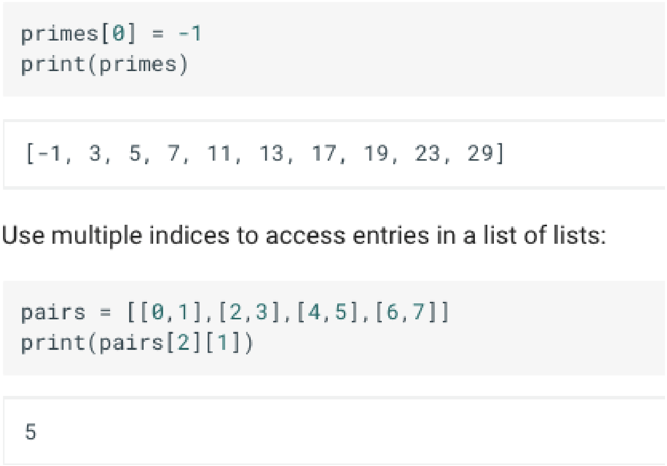
```
3
5
17
```

Use negative indices to access elements starting from the end of the list:

```
print(primes[-1])
print(primes[-2])
```

```
29
23
```

Since lists are mutable, we may assign new values to entries in a list:



```
primes[0] = -1
print(primes)
```

```
[-1, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

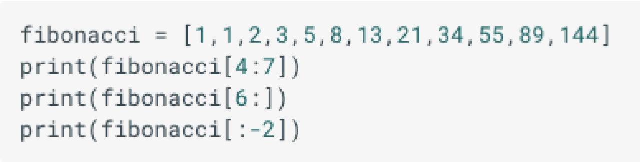
Use multiple indices to access entries in a list of lists:

```
pairs = [[0,1],[2,3],[4,5],[6,7]]
print(pairs[2][1])
```

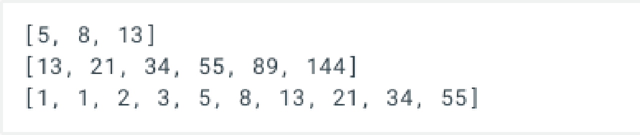
```
5
```

Slice

Create a new list form a sublist (called a slice):

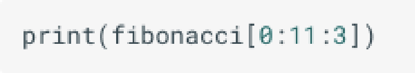


```
fibonacci = [1,1,2,3,5,8,13,21,34,55,89,144]
print(fibonacci[4:7])
print(fibonacci[6:])
print(fibonacci[:-2])
```

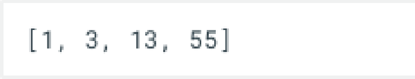


```
[5, 8, 13]
[13, 21, 34, 55, 89, 144]
[1, 1, 2, 3, 5, 8, 13, 21, 34, 55]
```

Notice in the example Fibonacci[4:7] the slice begins at index 4 and goes up to ***but not including*** index 7. A slice can skip over entries in a list. For example, create a slice from every third entry from index 0 to 11:



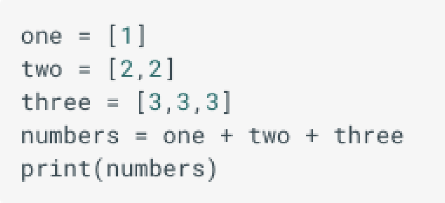
```
print(fibonacci[0:11:3])
```



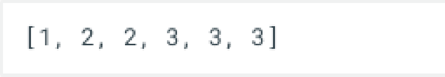
```
[1, 3, 13, 55]
```

Concatenate

The addition operator + concatenates lists:



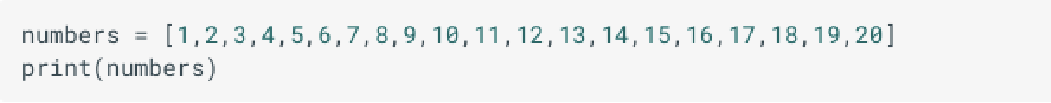
```
one = [1]
two = [2,2]
three = [3,3,3]
numbers = one + two + three
print(numbers)
```



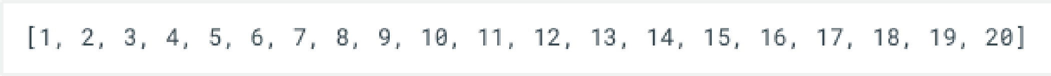
```
[1, 2, 2, 3, 3, 3]
```

List Comprehensions

The built-in function range() is an efficient tool for creating sequences of integers but what about an arbitrary sequence? It is very inefficient to create a sequence by manually typing the numbers. For example, simply typing out the numbers from 1 to 20 takes a long time!

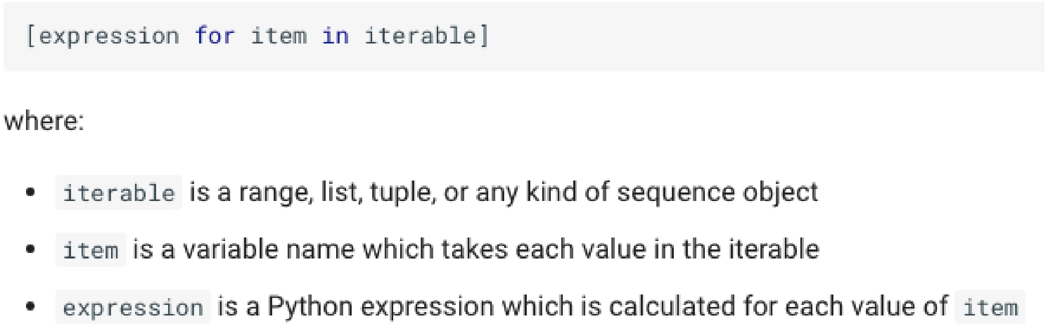


```
numbers = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
print(numbers)
```



```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```

Python has a beautiful syntax for creating lists called list comprehensions. The syntax is:

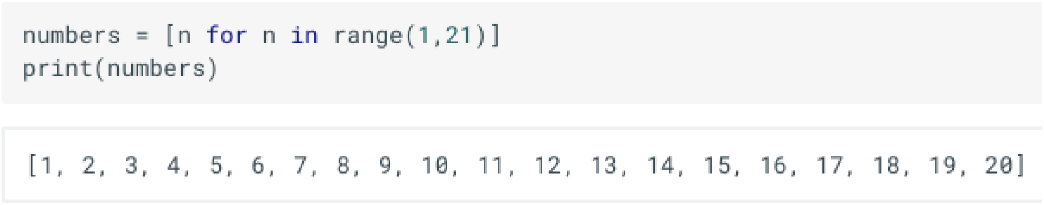


```
[expression for item in iterable]
```

where:

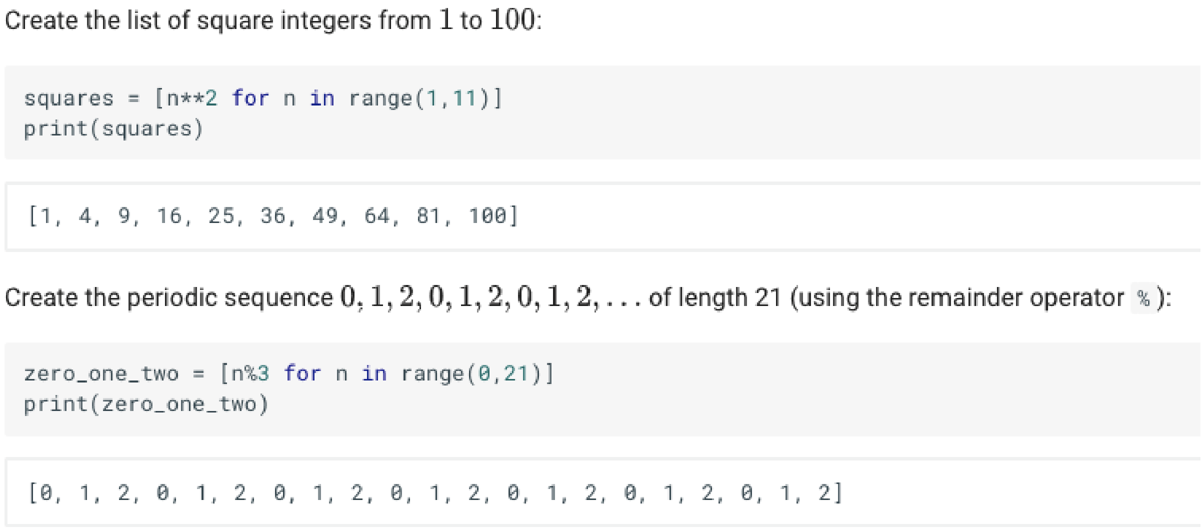
- `iterable` is a range, list, tuple, or any kind of sequence object
- `item` is a variable name which takes each value in the iterable
- `expression` is a Python expression which is calculated for each value of `item`

Use a list comprehension to create the list from 1 to 20:



```
numbers = [n for n in range(1,21)]
print(numbers)
```

```
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20]
```



Create the list of square integers from 1 to 100:

```
squares = [n**2 for n in range(1,11)]
print(squares)
```

```
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

Create the periodic sequence $0, 1, 2, 0, 1, 2, 0, 1, 2, \dots$ of length 21 (using the remainder operator `%`):

```
zero_one_two = [n%3 for n in range(0,21)]
print(zero_one_two)
```

```
[0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2, 0, 1, 2]
```

Practice Problem #1 Write a list comprehension to create a slice of lists:

[[0,0], [1,1], [2,4], [3,9], [4,16], [5,25], [6,36], [7,49]]

1.1 List of first three terms of this list.

1.2 List of the even-number index of this list.

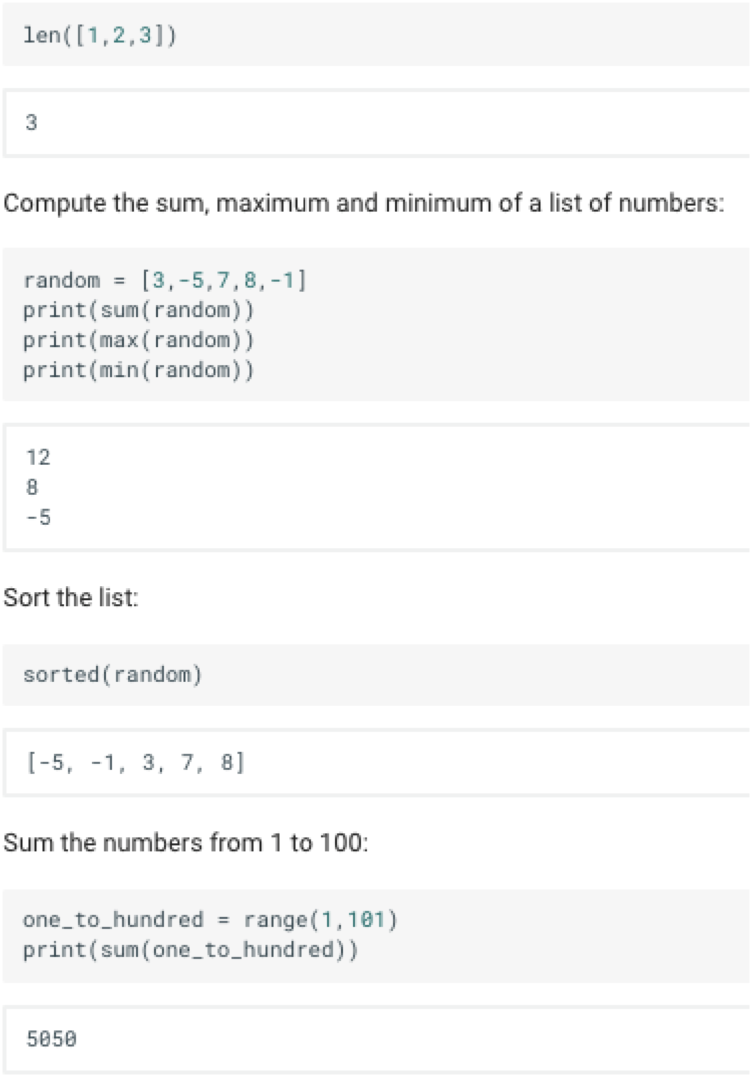
The outputs should look like these:

1.1 [[0,0], [1,1], [2,4]]

1.2[[0,0], [2,4], [4,16], [6,36]]

Built-in Functions for Sequences

Python has several built-in functions for computing with sequences. For example, compute the length of a list:



```
len([1,2,3])
```

```
3
```

Compute the sum, maximum and minimum of a list of numbers:

```
random = [3,-5,7,8,-1]
print(sum(random))
print(max(random))
print(min(random))
```

```
12
8
-5
```

Sort the list:

```
sorted(random)
```

```
[-5, -1, 3, 7, 8]
```

Sum the numbers from 1 to 100:

```
one_to_hundred = range(1,101)
print(sum(one_to_hundred))
```

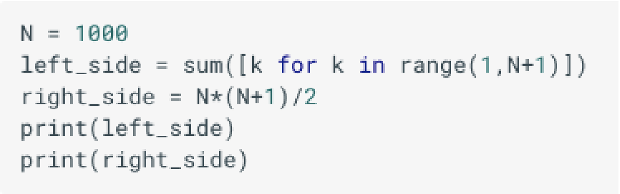
```
5050
```

Example 1.1: Triangle numbers

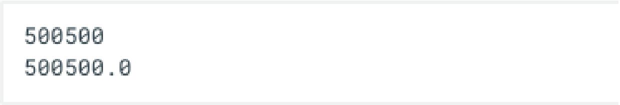
The formula for the sum of integers from 1 to N (also known as triangular numbers) is given by:

$$\sum_{k=1}^{N} k = \frac{N(N+1)}{2}$$

Let's verify the formula for N=1000:



```
N = 1000
left_side = sum([k for k in range(1,N+1)])
right_side = N*(N+1)/2
print(left_side)
print(right_side)
```



```
500500
500500.0
```

Notice the results agree (although the right side is a float since we used division).

Practice Problem #2 Determine the sum of squares (a special case of a geometric series) is given by the formula:

$$\sum_{k=1}^{N} k^2 = \frac{N(N+1)(2N+1)}{6}$$

Let’s verify the formula for N=2000.

2. Series

2.1 Finding the *first k terms* of a series:

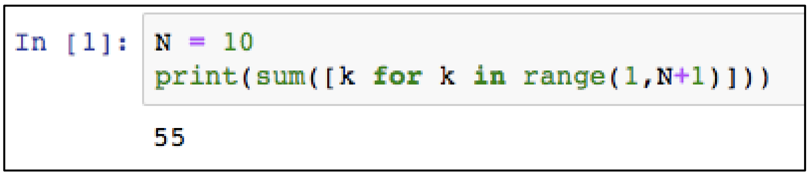
$$S_n = \sum_{k=1}^{n} a_k$$

The outputs are S_{k_1}, S_{k_1+1}, S_{k_1+2}…, S_{k_2} such that S_{k_1}= a_{k_1}, S_{k_1+1}= a_{k_1} + a_{k_2}, S_{k_1+2} = a_{k_1} + a_{k_2} + a_{k_3} , and so on.

Example 2.1 Consider the first 10 terms of the series:

$$S_n = \sum_{k=1}^{n} a_k$$

where $a_k = k$



```
In [1]: N = 10
        print(sum([k for k in range(1,N+1)]))

        55
```

<u>Practice Problem #3</u> Consider the sequence $a_n = \frac{1}{\sqrt{n}}$, where n = 1, 2, 3,…

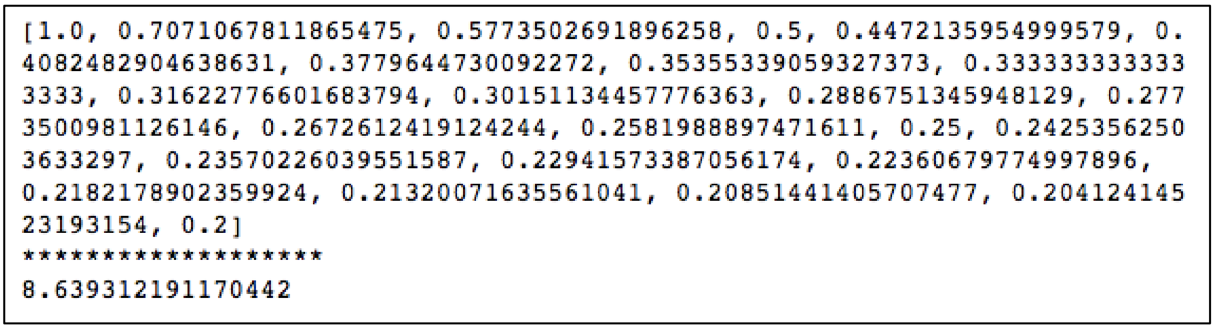
3.1 List the first 25 terms of the sequence a_n

3.2 Find the first 25 accumulative sums.

$$S_n = \sum_{k=1}^{n} a_k$$

Hint: Use Math library for square root

The outputs should look like these:



```
[1.0, 0.7071067811865475, 0.5773502691896258, 0.5, 0.4472135954999579, 0.
4082482904638631, 0.3779644730092272, 0.35355339059327373, 0.333333333333
3333, 0.31622776601683794, 0.30151134457776363, 0.2886751345948129, 0.277
3500981126146, 0.2672612419124244, 0.2581988897471611, 0.25, 0.2425356250
3633297, 0.23570226039551587, 0.22941573387056174, 0.22360679774997896,
0.2182178902359924, 0.21320071635561041, 0.20851441405707477, 0.204124145
23193154, 0.2]
*******************
8.639312191170442
```