This is a list of topics that may be included in Exam 1. This list is not exhaustive and is only intended to help you review.

Preparing for the exam:

1. *Study examples and activities.* Make sure you understand every examples and in-class activities.

2. *Read Python programs.* Try the programs from the lectures slides. For each program, go through each instruction and find the expected output. Next, run the same program on your computer and compare the output to your expected output. If there is a match, then you understand the program.

3. *Writing Python programs.* Pick some of the programs that we discussed during the lecture. Write a different solution for the problem. Remember, there is no one way to solve a problem.

4. *Study the lab problems.* Make sure you understand the lab problems and your solutions to the problem.
1 Topics

1. For loops
2. While loops
3. Lists
   a) Operation on lists
   b) List comprehension
4. Nested structures: nested loops, nested lists, nested sets
5. Sets
   a) Set operations such as union, intersection, difference, and issubset
   b) The add() and update() functions/methods
   c) How are sets, lists, and strings similar?
   d) How are sets, lists, and strings different?
6. join() operation: converts a list of strings into a string
7. Dictionaries
   a) Create a dictionary
   b) Add, change items in a dictionary
   c) Dictionary comprehension
   d) Use dictionary built-in functions
8. Randomness
   a) Module import
   b) Random module: common functions
9. Functions
   a) Functions: properties
   b) The __name__ special variable
   c) Built-in functions vs. methods vs. user-defined functions
   d) Local variables vs global variables
2 Some practice problems

Below are sample questions to help you prepare for the exam. Make sure you can solve all of these problems by hand. For most of the questions, you can check your answers by typing in the programs and seeing what happens on the computer.

1. What is the output of the following program. The symbol \texttt{ } represents a space.

```python
1 \texttt{a=“1,2,3”}
2 \texttt{b=“hi there!”}
3 \texttt{c=“4 10 19”}
4 \texttt{d=“45 150 1955”}
5 \texttt{e=“45:13:15:19”}
6
7 \texttt{print(a.split(“,”))}
8 \texttt{print(b.strip())}
9 \texttt{print(c.split())}
10 \texttt{print(d.split(“5”))}
11 \texttt{print(e.strip().split(“:”))}
```

a) What output does Line A produce?
b) What output does Line B produce?
c) What output does Line C produce?
d) What output does Line D produce?
e) What output does Line E produce?
2. Write a Python program that prompts the user for a number $n$ and calculates the factorial of that number $n$. The factorial of a positive integer $n$ called $n!$ is the product of all positive integers less than or equal to $n$. e.g. $5! = 5 \times 4 \times 3 \times 2 \times 1$.

$n$ should be positive otherwise the program should print an error message.

Sample output:

```
1 Enter a number: 5
2 Factorial: 120
```
3. What is the output of the following program.

```python
# Define a list of time lists
times = ["2:09", {"8:23", "3:32"}, {"2:52", "12:44", "12:02"}]
# Loop through each list
for day in times:
    print("day:", day)
    for time in day:
        print("time:", time)
    print()
```
4. Consider the following program.

```python
fruits = ["cherry", "strawberry rhubarb", "apple"]
print([fruit + " pie" for fruit in fruits])  # Line A
print([fruit.count("r") for fruit in fruits])  # Line B
print([fruit.find("a") for fruit in fruits])  # Line C
vec = [2, 4, 6]
print([3*x for x in vec if x > 3])  # Line D
print([[x, x**2] for x in vec])  # Line E
vec1 = [2, 4, 6]
vec2 = [4, 3, -9]
print([x*y for x in vec1 for y in vec2])  # Line F
print([vec1[i]*vec2[i] for i in range(len(vec1))])  # Line G
```

a) What output does Line A produce?
b) What output does Line B produce?
c) What output does Line C produce?
d) What output does Line D produce?
e) What output does Line E produce?
f) What output does Line F produce?
g) What output does Line G produce?
5. Write a program to compute \( \sum_{i=1}^{n} i = 1 + 2 + \ldots + n \), where \( n \) is specified by the user.

6. Write a program that asks the user to enter a list of integers and then prints a histogram to the screen. The integers are entered as a string separated with spaces. For your program, you can only use the material that we have discussed in class. See examples below.

**Example #1.** The user enters the string \(1 3 20 5\) (line 1), where the symbol \(\_\) represents a space. The output is then a histogram reflecting the values of the positive integers in the string (lines 3–6).

```
Enter a string of positive integers separated by spaces: 1 3 20 5

*  
***  
**************  
*****
```

**Example #2.** The user enters the string \(20 18 3 9 45\) (line 1), where the symbol \(\_\) represents a space. The output is then a histogram reflecting the values of the positive integers in the string (lines 3–6).

```
Enter a string of positive integers separated by spaces: 20 18 3 9 45

*******************  
*******************  
***  
*****  
************  
```


7. For an additional challenge, print the histogram vertically instead of horizontally.

Example.

```
Enter a string of positive integers separated by spaces: 1 3 20 5
```

```
* * * *
* * * 
* * *
* * *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
* *
```

8. A “wobbly” number is one in which the digits alternate between being higher and lower than the preceding one. Here are some wobbly numbers: 19284756242, 90909, 0909. Write a function called `wobbly(number)`, where number is an integer. If the number is wobbly, the function should return True. Otherwise, it returns False. For example,

- `wobbly(19284756242)` returns True.
- `wobbly(89012345)` returns False.
9. Write a program to print Pascal’s triangle, where the user specifies the number of rows in the triangle. On the outside are 1’s and each other number is the sum of the two numbers directly above it (see 11(b)).

a) Generate Pascal’s triangle, where the output is unformatted (not in a triangle). Give the user a prompt (such as Rows) for them to enter the number of rows of Pascal’s triangle to output.

Example.

```
Rows: 6
1
2
3
1 1
4
1 2 1
5
1 3 3 1
6
1 4 6 4 1
7
1 5 10 10 5 1
8
```

b) For an additional challenge, output the result in a triangle.

Example. The output is based on the user specifying that they want 6 rows of Pascal’s triangle printed.

```
Rows: 6
1
2
3
1 1
4
1 2 1
5
1 3 3 1
6
1 4 6 4 1
7
1 5 10 10 5 1
8
```

10. With a given integral number n, write a program to generate a dictionary that contains \((i, i \times i)\) such that is an integral number between 1 and n (both included). and then the program should print the dictionary. Suppose the following input is supplied to the program: 8 Then, the output should be:

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64}
```

11. Write a Python program to create and display all combinations of letters, selecting each letter from a different key in a dictionary.

```
Sample data : {"1":["a","b"], "2":["c","d"]}
```

Expected Output:

```
ac
ad
bc
bd
```