Lab-5

➢ Unbounded loops

Unbounded loops are used when the programmer doesn't know how many times the loop will be iterated. In unbounded loops, loop repeats and executed until the condition returns false.

```
i = 0
While True:
    if i== 4:
        break
    print('I am in the loop)
    i += 1
```

Makes the above code into bounded loop

```
for i in range(4):
    print('I am in the loop)
```

another example:

```
lst = ['a', 'b', 'c', 'd', 'f']
var = ''
i = 0
while True:
    var = lst[i]
    i+=1
    print(var)
    if var == 'd':
        break
```

Makes the above code into bounded loop

```
lst = ['a', 'b', 'c', 'd', 'f']
for item in lst:
    print(item)
    if item == 'd':
        break
```

another example: add even number to new list

```
lst = [4, 2, 1, 6, 7]
new_lst = []
while lst:
  if lst[0]%2==0:
    new_lst += lst[0]]
  lst = lst[1:]

return new_lst
```

using bounded loop instead:

```
lst = [4, 2, 1, 6, 7]
new_lst = []
for num in lst:
  if num%2==0:
    new_lst += [num]
return new_lst
```

- **<u>Lambda functions (anonymous functions):</u>**

In Python, an anonymous function or lambda function is a function that is defined without a name.

For normal functions we use keyword def in python, but for anonymous function we use lambda to define it.

You can think of lambda expressions as "inline function definitions." You don't even need to give these functions a name -- they are discarded immediately after use unless you store them in a variable!

```
>>> # Lambda
>>> (lambda x: x**2)(5)
25
>>> # We can also have multiple parameters
>>> (lambda x, y, z: x**2 + 2*y - z)(5, 6, 7)
30
>>> # Store lambda function in a variable
>>> summation = (lambda num1, num2: num1+num2)
>>> summation(3,5)
8
```

- **Modulo operator**

Modulo operator shown by %, it's an operator which return the remainder of integer division.

```
>>> num1 = 10
>>> num2 = 3
>>> remainder = num1%num2
>>> remainder
1
```

Integer division

When we use '/' operator in python 3, it returns the true value. i.e 3/2 = 1.5

But when we use '//' (floor division operator) operator in python 3, it rounds down to the nearest integer. i.e 3//2 = 1

```
>>> 5/2
2.5
>>> 5//2
2
>>> -5/2
-2.5
>>> -5//2
-3
```

➢ **<u>HW readme explanation with examples</u>**

We provide a2.py and the unit test result and explanation for that in the assignmentreadme.md. This is to make you all understand and interpret your readmes especially the unit test sections.

To run any of this code, you will all have to manually run it in the terminal (**in Visual Studio Code**).

One of 2 ways:

- run any python file, then type clear.
- Just type in python3 (Mac) / py (Windows)
- Press the up arrow and delete everything up to python and autocomplete the path.

Note: You all should not change it to a.py file

We will go through syntax.txt, type.txt, and errors.txt together.

The idea is to run each file repeatedly. Whenever an error comes up, your thought process should be able to diagnose it.

In the process, we will learn:

1. Python's error messages usually tell you exactly where to look, and
2. it's not always obvious what the source of the error is.