Uttara Python Lab 1

: q to exit the help
exit() to quit

help(<>)
dir(<>)

1. Print hello, print hello with concat of bye
2. Test all arithmetic op -> +,-,*,/,**,//,%
3. Test relational (<,>,..) & conditional (and/or) operators
4. Test compound assignment op (+=,-=,*=,/=,%=)
5. Test bitwise op (<<,>>,&,|,ˆ). Test 10 & 7, 10 | 7, 10 ^7, 20 << 1, 20 >> 2, 20 ˆ 15
6. Test Control Statements -> if, while, for syntax first
7. Create an integer variable, increment its value
8. Print multiplication tables of a num using both while & for
9. Print 1 to 10 to monitor
10. Print 10 to 1 to monitor
11. Print 1,3,5,7 till 50 to monitor
12. Print all values from 1 to 100 except multiples of 3
13. Print all values from 1 to 100 that are multiples of 3 or 7 but not both
14. Print all values from 1 to 100. Skip printing value if its divisible by 15 and exit the loop if value is divisible by 59. Use continue & break.

Using IDLE:

1. Try id() & type() with variables. Check whether pooling is done for integers & strings and for what range of values.
2. Try converting variables from integer, float, boolean to string & vice versa -> bool(),str(),int(),float(),list()
3. Create a function called test(). Accept a parameter. Print “in test” concatenated with the passed argument. Try to invoke by passing no argument & multiple arguments. Now do a help(test) and check. Add the annotation to describe what type of param to accept.
4. Create a function to print the passed argument 10 times to the monitor
5. Create a function to accept only an int as parameter (validate) and then print whether it is an odd/even number
6. Print all prime numbers from 1 to 1000 to monitor by coding a isPrime()
7. Print fibonacci series from 1 to 1000 to monitor
8. Create a function called test() that accepts an int. Create a variable named i = 10 and then pass i to test(i). Print the value of the argument in the test(). Now after invoking test(), print the i value. Is it the same? Then in the function, change the value of i and

then check outside the function whether the value has changed. Why not? Next pass a list, change its contents by invoking li.append(10) and then check after invoking whether the list has changed. Why so?

Example code:

First.py

```
def test(val:int)->int:
    ''' this is a test function that does nothing but print what is passed '''
    if isinstance(val,int):
        print('in test() '+str(val))
        val = val + 1
        return val
    else:
        print('invalid input')

print(test(10))
```

Second.py

```
def isPrime(num:int):
     if num==1:
          return False

     if num%2 == 0:
         return False

     if num == 3 or num == 5 or num == 7:
          return True

     for i in range(3, num//2):
         if num % i == 0:
             return False

     return True

print(isPrime(15))
print(isPrime(23))
print(isPrime(123123))
```

```
for x in range(1,1000,2):
    print(str(x) + ' is prime ? '+str(isPrime(x)))
```