311 – Numerical Computations
Lab 1: Introduction to Python (I/O, Mathematical Operators, Indentation and Conditionals)

Designed by: Guido van Rossum: A scientist from Netherlands (born 1956).

Released on 20 Feb 1991

1994 → Python 1

2000 → Python 2

2008 → Python 3: was a major revision of the language that is not completely <u>backward-compatible</u> and much Python 2 code does not run unmodified on Python 3.

1- Python is dynamically-typed language

```
y=3
y=False
y="welcome"
print(y)
```

You can change the type in run time

So each variable has a certain type at each step of the program

==========

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

```
x=3
y=x+z                    #z is not defined!!
```

NameError: name 'z' is not defined

2- It has no 'Character' type.

```
x=4
print(type(x))

x='Kuwait'
print(type(x))

x="Python"
print(type(x))

x='s'
print(type(x))

x="s"
print(type(x))

x=3.4
print(type(x))

x=True
print(type(x))
```

```
Output:
<class 'int'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'str'>
<class 'float'>
<class 'bool'>
```

3- Conditionals and
4- Python uses indentation to indicate a block of code

```
x=7
if x>3:
        print('hello')
        print('311')
if x%2 == 0:
        if x >100:
                print('www')
else:
        print('Odd')
        print('not even')
print('Bye')
```

==========

```
x=2
if  x==1:
        print('One')
elif  x==2:
        print('Two')
elif x==3:
        print('Three')
        print('Great')
else:
        print('Wrong')
print('Bye')
```

5- Input:

Python reads anything as a string (why?????)

```
x=input("Enter first Integer")          //assume 23
y=input("Enter Second Integer")         //assume 705
z = x + y
print("Result" , z)
```

output: !!!!!!

```
Result 23705
```

Correction:

```
x=input("Enter first Integer")
y=input("Enter Second Integer")
x=int(x)
y=int(y)
z = x + y
print("Result" , z)
```

or simply:

```
x= int(input("Enter first Integer"))
y= int(input("Enter Second Integer"))
```

Remark: the input( ) in python is a function that returns a string

6- Output

```
print("The quick" , "brown" , "fox jumps over the lazy" , "dog")

print("Bye")
```

Output:

```
The quick brown fox jumps over the lazy dog

Bye
```

```
print("The quick" , "brown" , "fox jumps over the lazy" , "dog", sep='#')

print("Bye")
```

Output:

```
The quick#brown#fox jumps over the lazy#dog

Bye
```

```
print("The quick" , "brown" , "fox jumps over the lazy" , "dog", end="3")

print("Bye")
```

Output:

```
The quick brown fox jumps over the lazy dog3Bye
```

Remark: You can use both (sep) and (end) in same print statement

7- Mathematical operations and Type casting

```
x= 7 / 2
print(x)              #prints 3.5

so how to make integer division??
x = 7 // 2
print(x)              #prints 3
Also:   x=3.5 // 0.4   →   x=8.0

or
x= int (7 / 2)
print(x)              #prints 3
==========================================
str(35)  →  '35'          float('4.25') → 4.25
int(3.8) → 3              bool(3) → True
bool(0) → False           float(4) → 4.0
==========================================

7 % 2  →  1          # the mod
==========================================
2 ** 3 →  8          # the power
4 ** 0.5 →  2
==========================================
```

8- Python is Interpreted

a- Consider this program:

```
x=float(input("Enter a number:")

z=x+y
```

The 1st statement will run then a run time error will occur in 2nd statement:

NameError: name 'y' is not defined

b- (this program will run successfully if x > y) !!

```
Import math
x= int(input("Enter first Integer"))
y= int(input("Enter Second Integer"))
if x > y:
        x +=  5
        print(math.sqrt(x))
else:
        y+=  12
        prnt(math.sqrt(x,y,z))        #A lot of errors (but not syntax)
```

9- Comparison Operators and Logical Operators:

Comparison:

==, !=, >, <, >=, <= **(very similar to the C family)**

Logical: and, or, not

Example: if x < y and not w % 7 == 3 :

However if you need to express: a is between x and y, you may directly use: x <= a <= y
(or: x < a < y, in case of non-inclusion).

- **Python Operator Precedence**

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift
&	Bitwise 'AND'
^ |	Bitwise exclusive `OR' and regular `OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

Exercise:

Q	Expression	Value
1	3 - - 2 * 6	
2	-2**2 + 1	
3	- 5 - - 3 - - 4	

Lab Task:

Write a Python Program that reads an integer.

If this integer is between 100 and 999 inclusively then the program uses basically the mod (%) operation, and other mathematical operations, to find the sum and product of that integer and its mirror image. Otherwise the error message: “Not composed of 3 digits” should appear.

Sample Run:

Enter an integer: 947

The sum of 947 and its mirror image is: 1696

The product of 947 and its mirror image is: 709303

Remark:

You can temporarily use the online compiler:

https://www.onlinegdb.com/online_python_compiler