Lecture 7 – Expressions and Arithmetic.

Python includes three data types for numbers: the data type **`int`** for integers or whole numbers, the data type **`float`** for floating-point numbers containing a decimal point, and the data type **`complex`** for complex numbers, which are beyond the scope of this course. Python also provides string support.

Arithmetic Expressions

Like most languages, Python specifies computation in the form of an arithmetic expression, which consists of terms joined by operators.

Arithmetic Operators

The following shows the most common arithmetic operators:

+	addition operator
—	subtraction operator
*	multiplication operator
/	division operator
//	integer divide operator
%	modulus (remainder) operator

Examples

```
>>> 5 + 3
8

>>> 6 - 1.8
4.2

>>> (4+1)*5
25

>>> 7/4
1.75

>>> 7//4
1

>>> 9/12
0.75

>>> 9//12
0

>>> 7 % 4
3

>>> 4 % 5
4

>>> 12 % 2
0

>>> 17 % 2
1
```

Relational Operators

Relational operators are used to **compare two numbers**. The following table shows the common relational operators:

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equal to
!=	Not equal to

Examples

```
>>> 6 > 3
True

>>> 4 < 7
True

>>> 3 == 3
True

>>> 3 != 4
True

>>> (5+3) >= 8
True

>>> 5 > 8
False

>>> (10-7) <= 2
False
```

Logical Operators

Logical operators are used to **combine** *Boolean expression* or **reverse** the value of a Boolean expression.

Operator	Description
not *x*	opposite of *x*
x **and** *y*	False unless *x* and *y* both True
x **or** *y*	True unless *x* and *y* both False

The **and** Operator's Truth Table

Expression	Evaluates to . . .
True and True	True
True and False	False
False and True	False
False and False	False

The or Operator's Truth Table

Expression	Evaluates to . . .
True or True	True
True or False	True
False or True	True
False or False	False

The not Operator

The not operator **operates on only one Boolean value** (or expression). This makes it a ***unary*** operator.

The not operator simply evaluates to the opposite Boolean value.

The not Operator's Truth Table

Expression	Evaluates to . . .
not True	False
not False	True

Examples

```
>>> not True
False

>>> not False
True

>>> True and True
True

>>> True and False
False

>>> False and True
False

>>> False and False
False

>>> True or True
True

>>> True or False
True

>>> False or True
True

>>> False or False
False

>>> (6 > 3) and (4 < 5)
True

>>> (3 > 0) or (1 > 10)
True
```

Operator precedence rule in Python

Operators	Meaning
()	Parentheses
**	Exponent
+x, -x, ~x	Unary plus, Unary minus, Bitwise NOT
*, /, //, %	Multiplication, Division, Floor division, Modulus
+, -	Addition, Subtraction
<<, >>	Bitwise shift operators
&	Bitwise AND
^	Bitwise XOR
|	Bitwise OR
==, !=, >, >=, <, <=, is, is not, in, not in	Comparisions, Identity, Membership operators
not	Logical NOT
and	Logical AND
or	Logical OR

What is the output of the following program

```
year = 2020

a = (((year % 4) == 0) and ((year % 100) != 0)) or ((year % 400) == 0)
b = (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
c = (year % 4 == 0 and year % 100 != 0) or year % 400 == 0
d = year % 4 == 0 and year % 100 != 0 or year % 400 == 0
e = ((year % 4 == 0) and (year % 100 != 0)) or (year % 400 == 0)

print(a)
print(b)
print(c)
print(d)
print(e)
```

Membership operators

Python has some special operators that have to do with **membership** of lists and tuples.

These are basically **used to determine if an object is a member of a list or tuple.**

Operator	Description
in	member of
not in	not a member of

```
>>> mylist = [1,3,5]

>>> 1 in mylist
True

>>> 2 in mylist
False

>>> 2 not in mylist
True

>>> mylist2 = ["cat","dog","horse"]
>>> "cat" in mylist2
True
```

THE DIFFERENCE BETWEEN THE == AND = OPERATORS

- The == operator **(equal to)** asks whether two values are the same as each other.

- The = operator **(assignment)** puts the value on the right into the variable on the left.

Accepting Data from the keyboard

There are hardly any programs without any input. Input can come in various ways, for example from a database, another computer, mouse clicks and movements or from the internet. Yet, in most cases the input stems from the keyboard.

For this purpose, Python provides the function **input().** Input has an optional parameter, which is the prompt string.

If the input function is called, the program flow will be stopped until the user has given an input and has ended the input with the return key. The text of the optional parameter, i.e. the prompt, will be printed on the screen.

Example

```
name    = input("Enter Employee Name")
salary  = input("Enter salary")
company = input ("Enter Company name")
print("Printing Employee Details")
print ("Name", "Salary", "Company")
print (name, salary, company)
```

Sample Run

```
Enter Employee Name John
Enter your salary 8000
Enter Company name Google
Printing Employee Details
Name Salary Company
John  8000  Google

>>>
```

Example

```
first_number = int ( input ("Enter first number    ") )
second_number = int ( input ("Enter second number  ") )

sum = first_number + second_number

print("Addition of two number is: ", sum)
```

Sample Run

```
Enter first number 12
Enter second number 14
Addition of two number is:  26

>>>
```

Example

```
float_number = float (input("Enter a float number  ") )
print ("input float number is: ", float_number )


Enter a float number  3.14
input float number is:  3.14
>>>


Enter a float number  5
input float number is:  5.0
>>>
```

Enter a float number a

```
Traceback (most recent call last):
  File "C:\Users\HP\AppData\Local\Programs\Python\Python38-
32\HelloWorld.py", line 10, in <module>
    float_number = float (input("Enter a float number") )
ValueError: could not convert string to float: 'a'
>>>
```

Get multiple values from the user in one line

```
name, age, phone = input("Enter Your Name, Age ,
Phone Number separated by space: ").split()

print("User Details: ", name, age , phone)
```

Sample Run

```
Enter your name, Age separated by space: abc 30 123456
User Details:  abc 30 123456
>>>
```

Formatting

Python print 2 decimal places :

In Python, to print 2 decimal places use

str.format() with “{:.2f}” as string and float as a number.

Call print and it will print the float with 2 decimal place

Example

```
float = 2.158327
format_float = "{:.2f}".format(float)
print(format_float)
```

Output

2.16

Another Way

```
x = 23.45741390245849
x = format(x, '.2f')
print ( x )
```

Output

```
23.46
```

Python float precision truncate :

Firstly, must **import** math module.

Then the **trunc()** function is used to remove all decimal parts from the floating-point numbers and it returns only the integer part of the number.

Example:

```
import math
float = 12.16785
print("The value of number is: ",end="")
print (math.trunc(float))
```

Output

```
The value of number is: 12
```

Python float precision ceil

In Python, the ceil() function is used to return the ceiling value of a number.

The ceil value is the **smallest integer** greater than that number.

Example:

```
import math
float = 12.16785
print("The smallest integer greater than number is: ",end="")
print (math.ceil(float))
```

Output

```
The smallest integer greater than number is: 13
```

Python float precision floor

Python floor() function is used to return the floor value of a number.

The floor value is the **greatest integer smaller than the number.**

Example:

```
import math
float = 12.16785
print("The greatest integer smaller than number is:
",end="")
print (math.floor(float))
```

Output

```
The greatest integer smaller than number is:  12
```

Python decimal format

To format decimals, use **str.format(number)** where a string is ‘{0:.3g}’ and it will format string with a number.

It will display the number with 1 number before the decimal and up to 2 numbers after the decimal.

Example:

```
number = 1.345
f = '{0:.3g}'.format(number)
print(f)
```

Output

1.34

Python round numbers

To round numbers in python, use the **round()** function.

The round function will round off a number to a given number of digits and makes rounding of numbers easier.

Example:

```
number = 1.345
print(round(number))
```

Output

1

Example:

```
number = 1.5
print(round(number))
```

Output

2

Limiting floats to two / three decimal points

The **round() function** returns a floating-point number which will be rounded to specified numbers, and it will round the float to two decimal places.

Example:

```
my_float = 2.13456
limit_float = round(my_float, 2)
print(limit_float)
```

output

2.13

Example:

```
my_float = 2.13456
limit_float = round(my_float, 3)
print(limit_float)
```

output

2.135

Python Modules

Math Module

Some of the most popular mathematical functions are defined in the math module. These include trigonometric functions, representation functions, logarithmic functions, angle conversion functions. **To import the math module use :**

```
>>> import math
```

To display all modules do

```
>>> dir(math)
```

['__doc__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2', 'atanh', 'cbrt', 'ceil', 'comb', 'copysign', 'cos', 'cosh', 'degrees', 'dist', 'e', 'erf', 'erfc', 'exp', 'exp2', 'expm1', 'fabs', 'factorial', 'floor', 'fmod', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'isqrt', 'lcm', 'ldexp', 'lgamma', 'log', 'log10', 'log1p', 'log2', 'modf', 'nan', 'nextafter', 'perm', 'pi', 'pow', 'prod', 'radians', 'remainder', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc', 'ulp']

>>>

Examples :

```
>>>math.pi
3.141592653589793

>>> math.pow(5,3)
125.0

>>> math.sqrt(100)
10.0

>>> math.fabs(-100)
100.0

>>> math.factorial(5)
120

>>> math.sin(100)
-0.5063656411097588
>>>
```

Random Number Generator

Sometimes we want the computer to pick a random number in a given range, pick a random element from a list, pick a random card from a deck, flip a coin, etc. The random module provides access to functions that support these types of operations. The `random` module is another library of functions that can extend the basic features of python. Other modules we have seen so far are `string`, `math`, `time` and `graphics`

>>> import random

>>> dir(random)

['BPF', 'LOG4', 'NV_MAGICCONST', 'RECIP_BPF', 'Random', 'SG_MAGICCONST', 'SystemRandom', 'TWOPI', '_Sequence', '_Set', '__all__', '__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', '_accumulate', '_acos', '_bisect', '_ceil', '_cos', '_e', '_exp', '_inst', '_log', '_os', '_pi', '_random', '_repeat', '_sha512', '_sin', '_sqrt', '_test', '_test_generator', '_urandom', '_warn', 'betavariate', 'choice', 'choices', 'expovariate', 'gammavariate', 'gauss', 'getrandbits', 'getstate', 'lognormvariate', 'normalvariate', 'paretovariate', 'randint', 'random', 'randrange', 'sample', 'seed', 'setstate', 'shuffle', 'triangular', 'uniform', 'vonmisesvariate', 'weibullvariate']

Examples

```
# Random float x, 0.0 <= x < 1.0
>>> random.random()
0.37444887175646646


# Random float x, 1.0 <= x < 10.0
>>> random.uniform(1, 10)
1.1800146073117523


# Integer from 1 to 10, endpoints included
>>> random.randint(1, 10)
7

# Even integer from 0 to 100
>>> random.randrange(0, 101, 2)
26


# Choose a random element
>>> random.choice('abcdefghij')
'c'

# Shuffle elements

>>> items = [1, 2, 3, 4, 5, 6, 7]
>>> random.shuffle(items)
>>> print(items)
[7, 3, 2, 5, 6, 4, 1]


# Choose 3 elements
>>> random.sample([1, 2, 3, 4, 5],  3)
[4, 1, 5]
```

Python System Module : sys

The sys module provides functions and variables used to manipulate different parts of the Python runtime environment. Must import the **sys module**

Examples :

sys.exit : This causes the script to exit back to either the Python console or the command prompt.

This is generally used to safely exit from the program in case of generation of an exception.

```
import sys


print("\nIn order to terminate any Python program , use system exit command")
print("\nBut first, you need to import the sys Module")
print("\nAny Statement after the exit command will not be executed")
print("\nThis is an example of terminating a python program\n")
sys.exit('message such as listofitems not long enough\n')
print("OK")
```

Sample Run

In order to terminate any Python program , use system exit command
But first, you need to import the sys Module
Any Statement after the exit command will not be executed
This is an example of terminating a python program

SystemExit: message such as listofitems not long enough

>>>

sys.maxsize : Returns the **largest** integer a variable can take.

```
>>> sys.maxsize

9223372036854775807
```

sys.version : This attribute displays a string containing the **version number** of the current Python interpreter.

```
>>> sys.version

'3.11.1 (tags/v3.11.1:a7a450f, Dec  6 2022, 19:58:39) [MSC v.1934 64 bit (AMD64)]'

>>>
```

```
>>> sys.copyright

'Copyright (c) 2001-2022 Python Software Foundation.\nAll Rights Reserved.\n\nCopyright (c) 2000 BeOpen.com.\nAll Rights Reserved.\n\nCopyright (c) 1995-2001 Corporation for National Research Initiatives.\nAll Rights Reserved.\n\nCopyright (c) 1991-1995 Stichting Mathematisch Centrum, Amsterdam.\nAll Rights Reserved.'

>>>
```

>>> dir(sys)

['__breakpointhook__', '__displayhook__', '__doc__', '__excepthook__', '__interactivehook__', '__loader__', '__name__', '__package__', '__spec__', '__stderr__', '__stdin__', '__stdout__', '__unraisablehook__', '_base_executable', '_clear_type_cache', '_current_frames', '_debugmallocstats', '_enablelegacywindowsfsencoding', '_framework', '_getframe', '_git', '_home', '_xoptions', 'addaudithook', 'api_version', 'argv', 'audit', 'base_exec_prefix', 'base_prefix', 'breakpointhook', 'builtin_module_names', 'byteorder', 'call_tracing', 'callstats', **'copyright'**, 'displayhook', 'dllhandle', 'dont_write_bytecode', 'exc_info', 'excepthook', 'exec_prefix', 'executable', **'exit'**, 'flags', 'float_info', 'float_repr_style', 'get_asyncgen_hooks', 'get_coroutine_origin_tracking_depth', 'getallocatedblocks', 'getcheckinterval', 'getdefaultencoding', 'getfilesystemencodeerrors', 'getfilesystemencoding', 'getprofile', 'getrecursionlimit', 'getrefcount', 'getsizeof', 'getswitchinterval', 'gettrace', 'getwindowsversion', 'hash_info', 'hexversion', 'implementation', 'int_info', 'intern', 'is_finalizing', **'maxsize'**, 'maxunicode', 'meta_path', 'modules', **'path'**, 'path_hooks', 'path_importer_cache', 'platform', 'prefix', 'pycache_prefix', 'set_asyncgen_hooks', 'set_coroutine_origin_tracking_depth', 'setcheckinterval', 'setprofile', 'setrecursionlimit', 'setswitchinterval', 'settrace', 'stderr', 'stdin', 'stdout', 'thread_info', 'unraisablehook', **'version'**, 'version_info', 'warnoptions', 'winver']