INTRODUCTION TO PYTHON

MPHYCC-05: Modeling and Simulation
Unit II: Introduction to Python Programming

By

Dr. Santosh Prasad Gupta
Department of Physics
Patna University
Patna

Lecture I

LOW LEVEL AND HIGH LEVEL LANGUAGE

- High level language and low level language are the programming languages's types.

- High level: programmers can easily understand or interpret or compile in comparison of machine language: Examples of high level languages are C, C++, Java, Python, etc.

- Low level: Machine can easily understand the low level language in comparison of human beings.

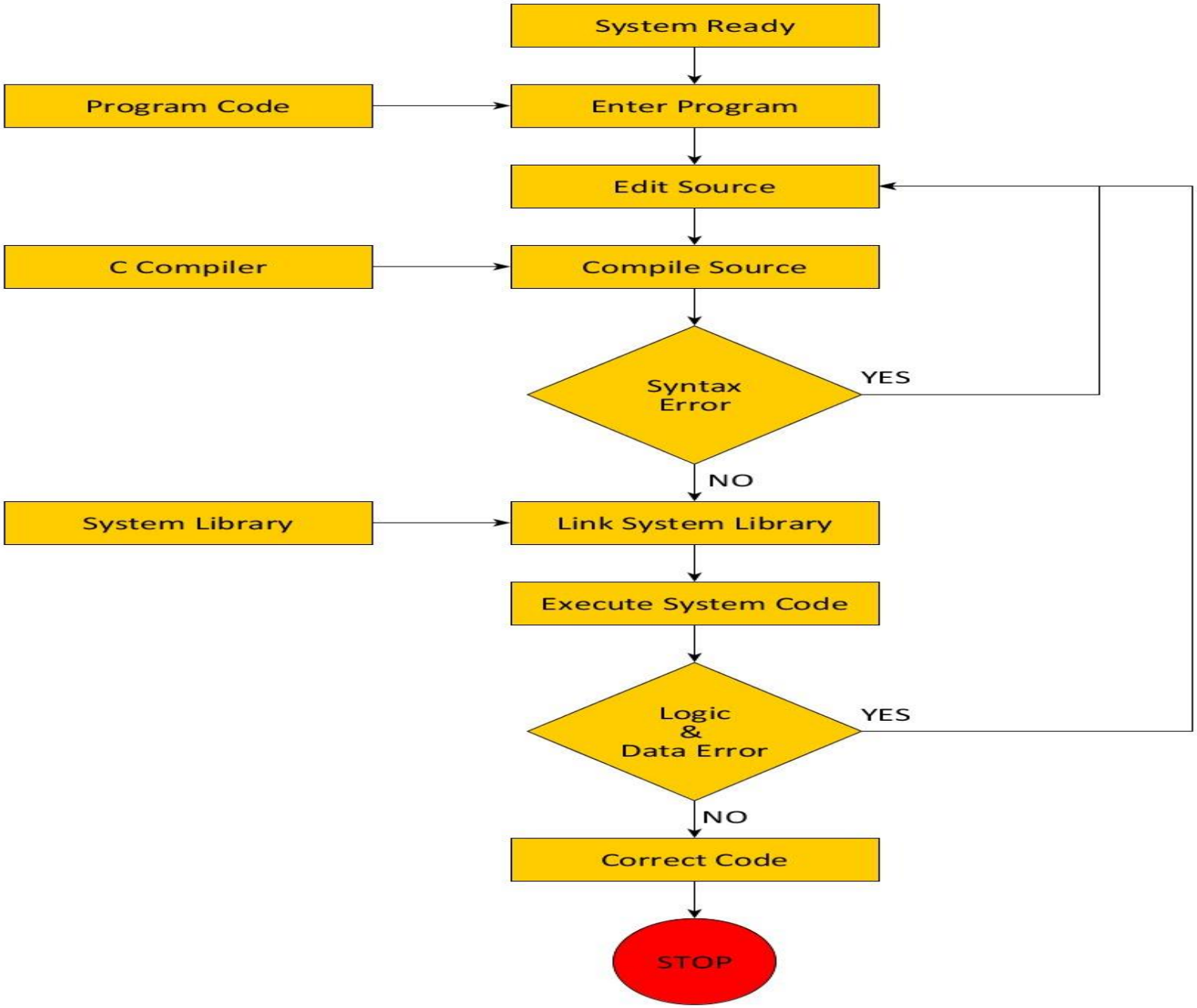
- Low-level languages can convert to machine code without a compiler or interpreter – second-generation programming languages use a simpler processor called an assembler.
  Example: assembly and machine code

COMPILER

- A compiler is a computer program that translates computer code written in one programming language (the source language) into another language (the target language).

- The name compiler is primarily used for programs that translate source code from a high-level programming language to a lower level language (e.g., assembly language, object code, or machine code) to create an executable program

FLOW CHART FOR COMPILING AND RUNNING A PROGRAMME



HISTORY OF PYTHON

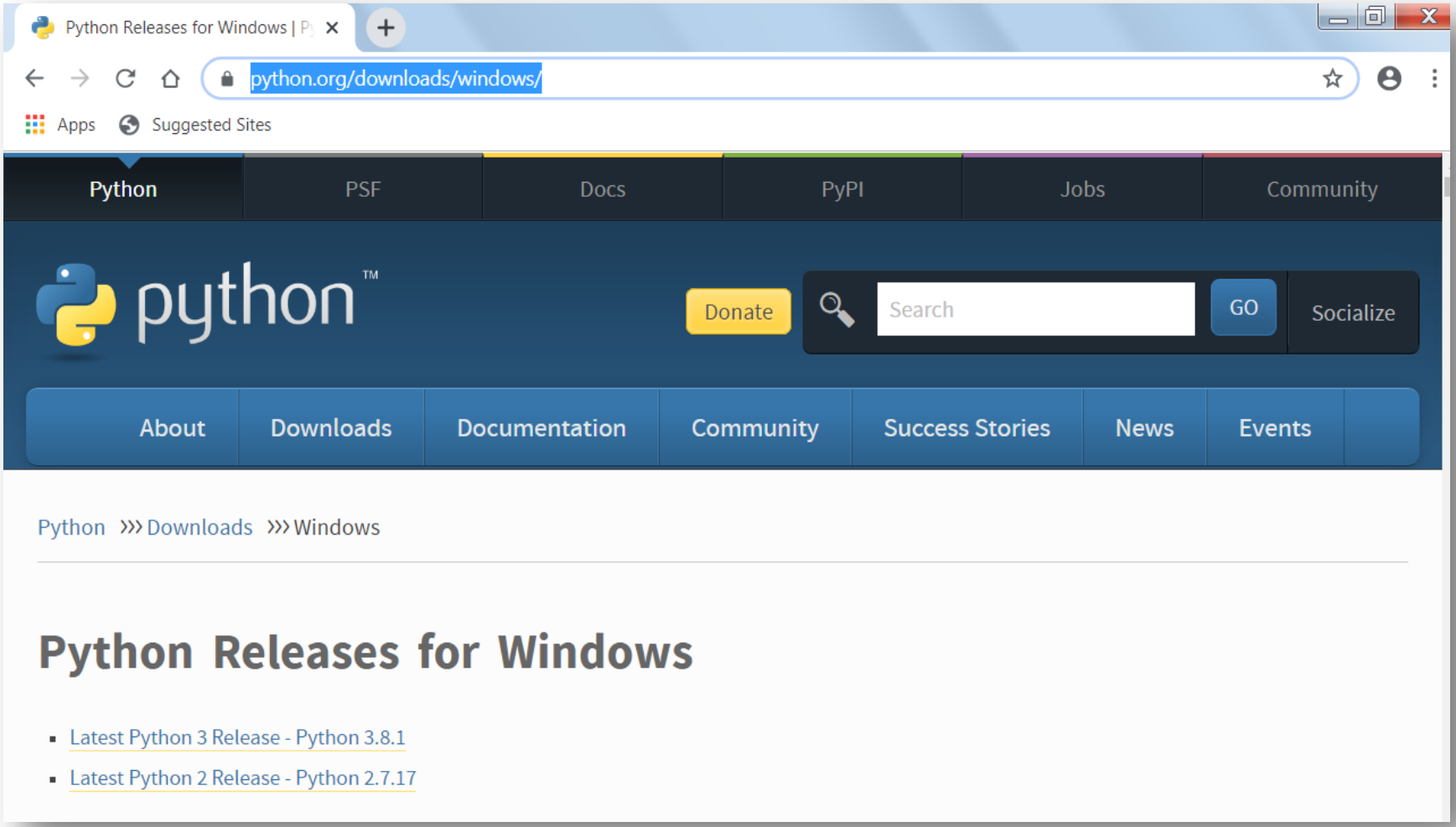
- Created in 1989 by Guido van Rossum
  - Created as a scripting language for administrative tasks
  - Based on *All Basic Code (ABC)* and *Modula-3*
    - Added extensibility
  - Named after comic troupe Monty Python

- Released publicly in 1991
  - Growing community of Python developers
  - Evolved into well-supported programming language

- Python is high level language

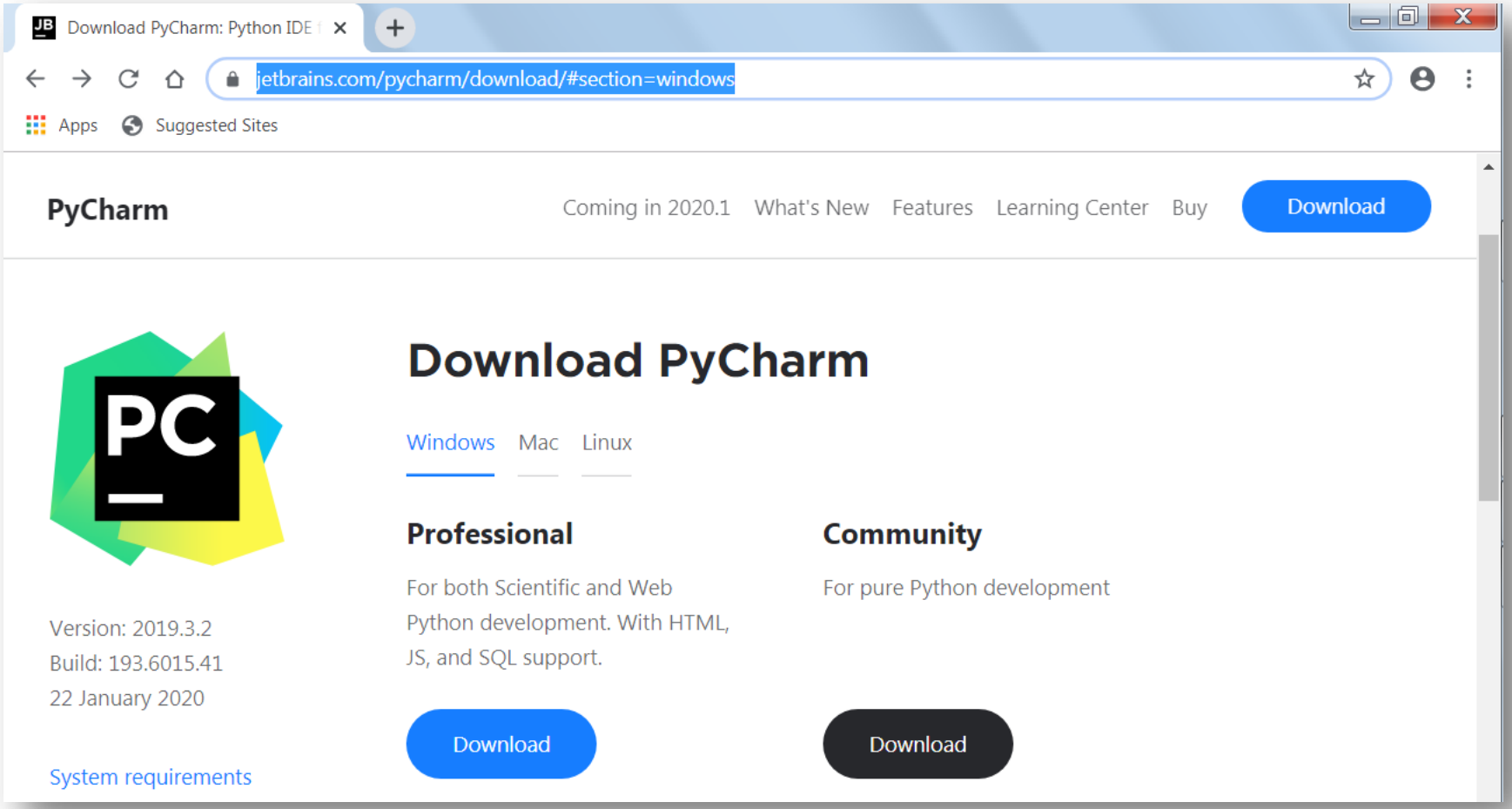
INSTALLING PYTHON

- Download the software from the site:
- https://www.python.org/downloads/windows/



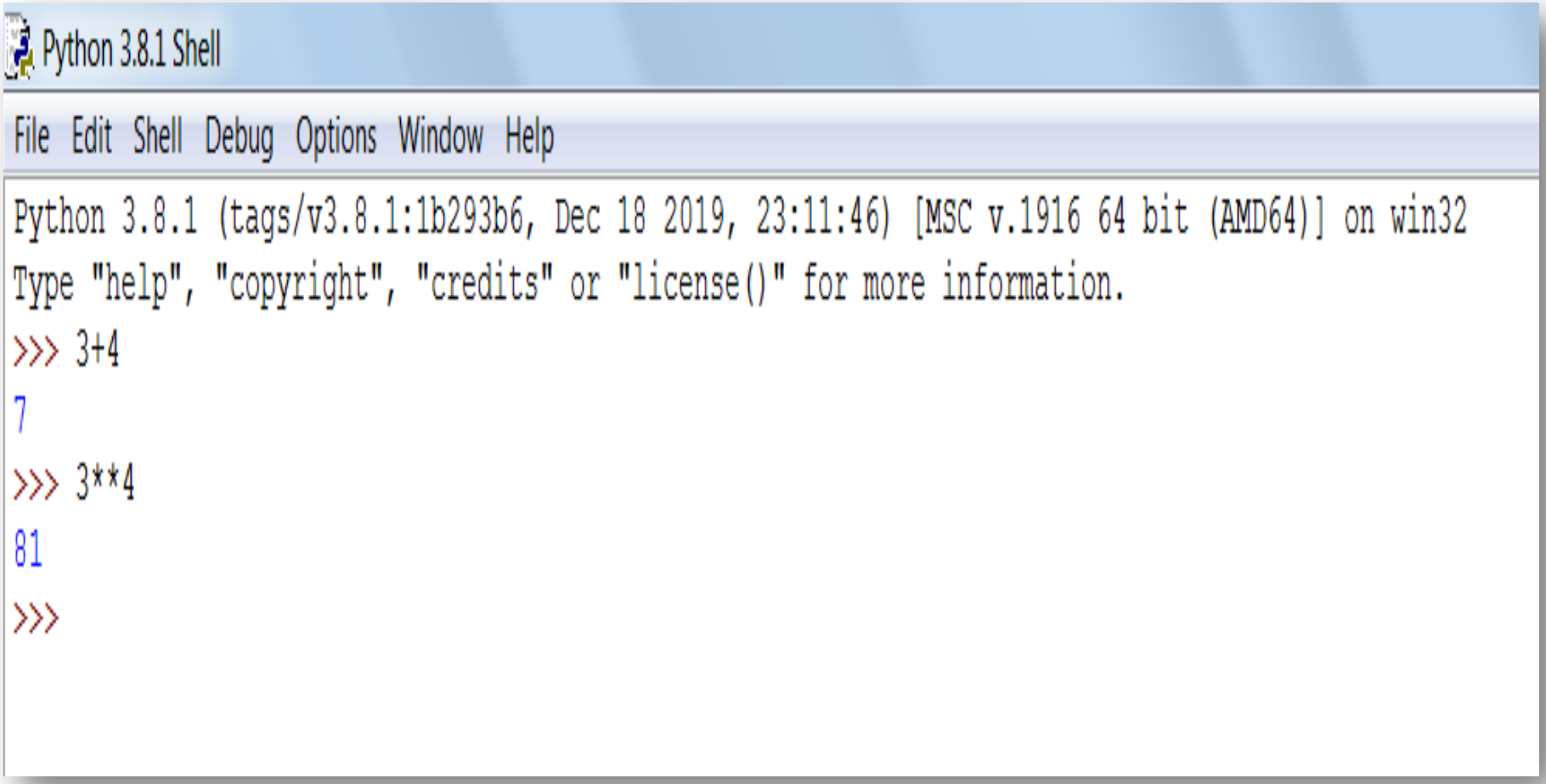
INSTALLING PYTHON: PYCHARM

- Download the software from the site:
- https://www.jetbrains.com/pycharm/download/#section=windows



PROGRAMMING IN PYTHON

- IDLE Interactive Shell: simple integrated development environment (IDE) that comes with Python. It's a program that allows you to type in your programs and run them



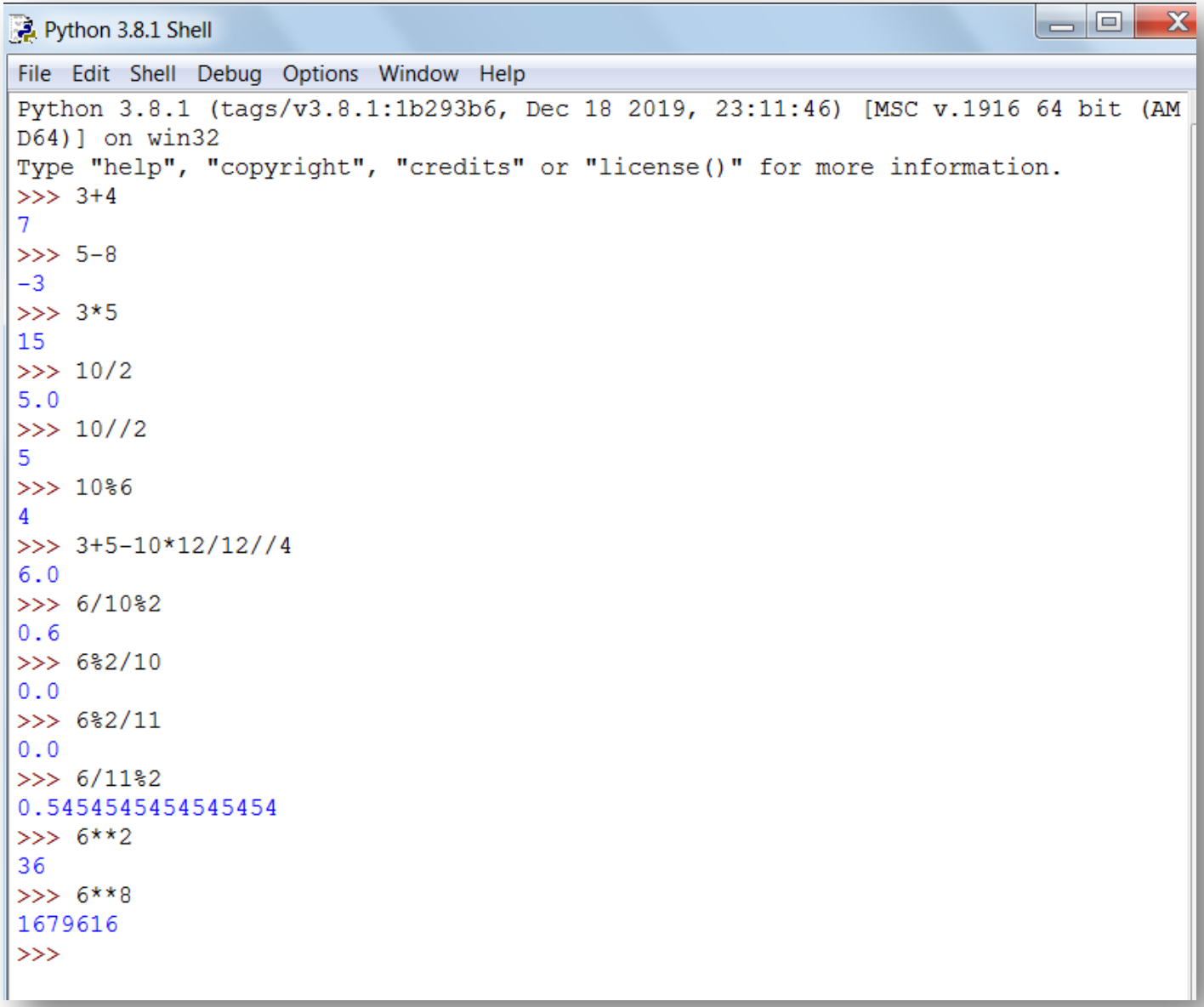
MATH OPERATOR

Operator	Description
+	addition
–	subtraction
*	multiplication
/	division
**	exponentiation
//	integer division
%	modulo (remainder)

ORDER OF OPERATION

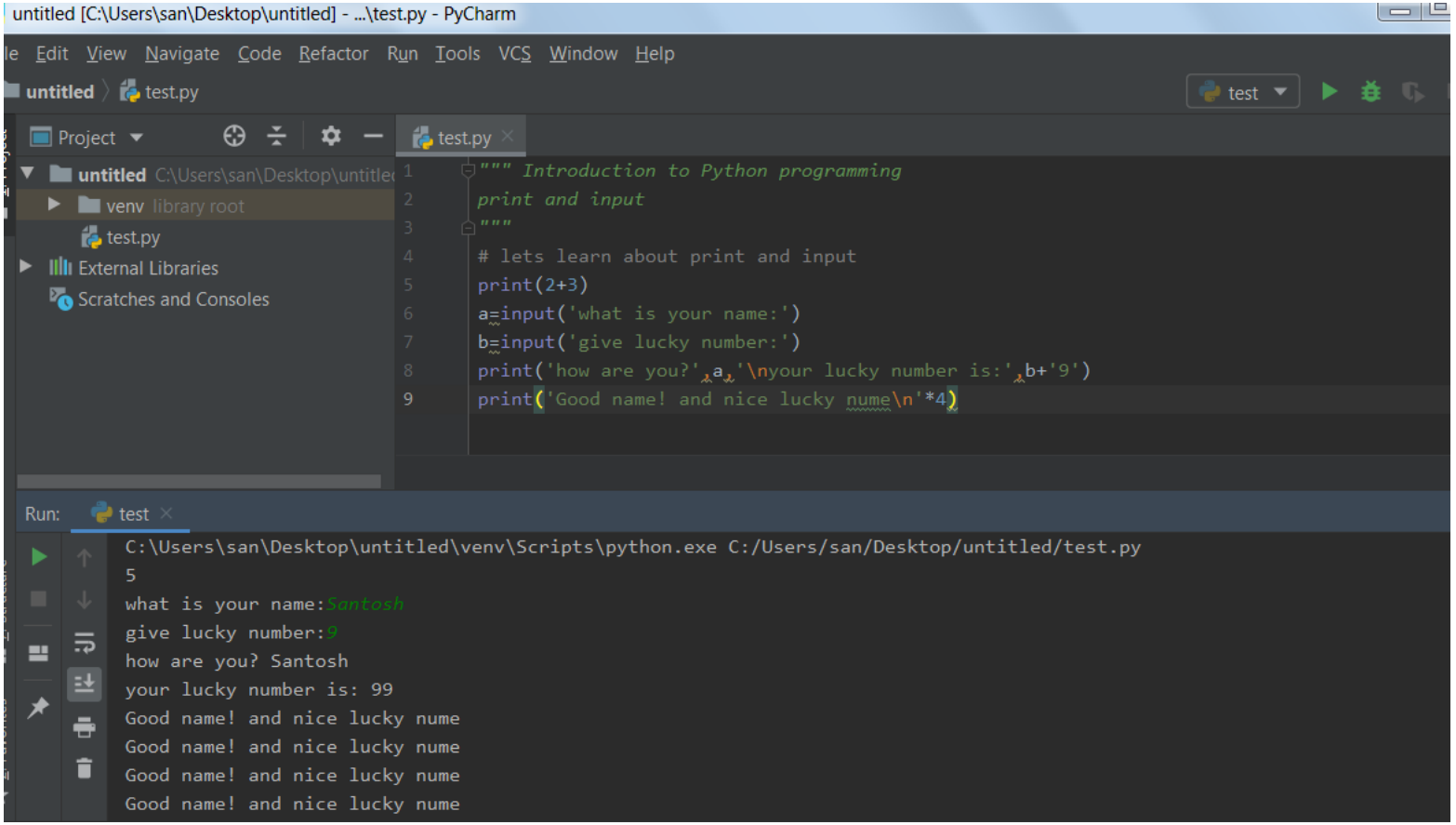
- Exponentiation gests first, followed by multiplication and division (including // and %) and addition and subtraction come last

PYTHON AS A CALCULATOR



COMMENT, PRINT, INPUT

- #..... For commenting single line
- ‘ ‘ ‘.....
  .....
  ..... ’ ’ ’ commenting multiple line



MORE ON input AND print

```
a = eval(input('Enter the value of first number:'))
b=eval(input('Enter the value of second number:'))
print(a*b,'----',a+c,'----',b+c)
```

```
a = eval(input('Enter the value of first number:'))
b=input('Enter the value of second number:'))
print('----',a+b,'----')
```

- \n new line
- \t tab
- \' for printing '
- \" for printing "

- print ('-----\t'*5)
- print('-----\t\t\t'*5)
- print('-----\n'*5)
- print('----\"*5)
- print('----\'"*5)
- print('----%'*5)
- print('--- # % & " '*5)

Lecture II

- for loop
- if
- elif
- while

for loop

- Probably the most powerful thing about computers is that they can repeat things over and over very quickly.
- There are several ways to repeat things in Python, the most common of which is the for loop.

```
# print hello in ten times
for i in range(10):
    print('Hello')
```

```
# print hello in ten times
for i in range(10):
    print('Hello',end= '' )
```

```
print('A')
print('B')
for i in range(5):
        print('C')
        print('D')
print('E')
print( 'loop is also over')
```

- The value we put in the **range** function determines how many times we will loop.
- The way **range** works is it produces a list of numbers from zero to the value minus one. For instance, **range**(5) produces five values: 0, 1, 2, 3, and 4.

range

```
Statement Values generated
range(10) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
range(1, 10) 1, 2, 3, 4, 5, 6, 7, 8, 9
range(3, 7) 3, 4, 5, 6
range(2, 15, 3) 2, 5, 8, 11, 14
range(9, 2, -1) 9, 8, 7, 6, 5, 4, 3
```

Q. Write a program that prints out a list of the integers from 1 to 20 and their squares. The output should look like this:

1 --- 1
2 --- 4
3 --- 9
...
20 --- 400

```
for i in range(1, 21):
    print(i,'---', i*i)
```

```
# try this
for i in range(1, 21):
    print( '*' *i)
```

MULTIPLICATION TABLE: NESTED FOR LOOP

```
for i in range(1,11):
    for j in range(1,11):
        print((i*j),end=' ')
    print()
# try putting more print()
```

```
for i in range(1,11):
    for j in range(1,11):
        print('{:3d}'.format(i*j),end=' ')
    print()
# try putting more print()
```

if **STATEMENT**

- **if** statement: when we only want to do something provided something else is true

Conditional operators

The comparison operators are ==, >, <, >=, <=, and !=. That last one is for not equals. Here are a few examples:

```
Expression Description
if x>5: if x is greater than 5
if x>=5: if x is greater than or equal to 5
if x==5: if x is 5
if x!=5: if x is not 5
```

There are three additional operators used to construct more complicated conditions: **and**, **or**, and **not**

- **Order of operations: and** is done before **or**, so if you have a complicated condition that contains both, you may need parentheses around the **or** condition.

```
a=eval(input('Enter your marks:'))
if a>=60 and a<=80:
    print('your grade is B')
```

```
a=eval(input('Enter your marks:'))
if a>=60 or a<=80:
    print('your grade is B')
```

```
a=eval(input('Enter your marks:'))
if a!=60 or a!=80:
    print('your grade is B')
```

```
marks = eval(input('Enter your score: '))
if marks >=90:
    print('A')
if marks >=80 and marks<90:
    print('B')
if marks >=70 and marks<80:
    print('C')
if marks >=60 and marks<70:
    print('D')
if marks <60:
    print('F')
```

elif STATEMENT

```
marks = eval(input('Enter your score: '))
if marks >=90:
    print('A')
elif marks >=80:
    print('B')
elif marks >=70:
    print('C')
elif marks >=60:
    print('D')
else:
   print('F')
```

while STATEMENT

```
count = 0
while (count < 9):
   print('The count is:', count)
   count = count + 1
print('bye!')
```

```
var = 1
while var == 1: # This constructs an infinite loop
  num=eval(input('Enter a number :'))
  print('You entered:', num)
print('Good bye!')
```

Getting help from Python

- There is documentation built into Python known as module
- Example: Python has a module called math that contains familiar math functions, including sin, cos, tan, exp, log, log10, factorial, sqrt

```
help()
help( 'module' )
import math
help(math)
from math import sin, pi
print(sin(pi/2))
```

Working with random

```
import random
print(random.random())          # Random float x, 0.0 <=
                                        x < 1.0
print(random.uniform(1, 10))    # Random float x, 1.0 <=
                                        x < 10.
print(random.randint(1, 10))    # random integer from 1
                                    to 10, endpoints included
print(random.randrange(0, 101, 3))  # integer from 0 to
                                      100, divided by three
print(random.choice('abcdefghij'))  # Choose a random element

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

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

More with with random

```
import random
for i in range(100):
    print(random.random())
```

Strings

- Strings are a data type in Python for dealing with text
- A string is created by enclosing text in quotes.
  - either single quotes, ', or
  - double quotes, ".
  - A triple-quote can be used for multi-line strings.

```
s =  'Hi How are You?'
t =  "Please go through it"
m = """This is a long string that is
spread across two lines.""" "
print(s,'\n',t,'\n',m)
```

```
num = eval(input('Enter a number: '))
string = input('Enter a string: ')
```

- The empty string ‘ ’ is the string equivalent of the number 0. It is a string with nothing in it.

- Length of a string (how many characters it has), use the built-in function **len**. For example, **len**('Hello') is 5.
- The operators + and * can be used on strings.

```
string = input('Enter a string: ')
print(len(string))
print( ‘AB’ + ‘CD’ )          ABCD
print( ‘Hi’ *4)               HiHiHiHi
print( ‘A’ + ‘7’ + ‘B’ )      A7B
```

```
s = ''
for i in range(10):
    t = input('Enter a letter: ')
    if t=='a' or t=='e' or t=='i' or t=='o' or t=='u':
        s = s + t
print(s)
if 'a' in s:
    print('Your string contains the letter a.')
else:
    print('a is not contained in your string')
```

Indexing: Python uses square brackets to index. The table below gives some examples of indexing the string s='Python'.

Statement	Result	Description
s[0]	P	first character of s
s[1]	y	second character of s
s[-1]	n	last character of s
s[-2]	o	second-to-last character of s

A slice is used to pick out part of a string.
s='abcdefghij'.
index: 0 1 2 3 4 5 6 7 8 9
letters: a b c d e f g h i j

Code	Result	Description
s[2:5]	cde	characters at indices 2, 3, 4
s[:5]	abcde	first five characters
s[5:]	fghij	characters from index 5 to the end
s[-2:]	ij	last two characters
s[:]	abcdefghij	entire string
s[1:7:2]	bdf	characters from index 1 to 6, by twos
s[: :-1]	jihgfedcba	a negative step reverses the string

```
s='abcdefghij'
print(s[0],s[1],s[2],s[-1],s[5])
print(s[2:7],'\n',s[:5], '\n',s[: : -1])
```

Strings come with a ton of **methods**, Here are some of the most useful ones: Method Description
lower() returns a string with in lowercase
upper() returns a string with in uppercase
replace(x,y) returns a string with x replaced by y
count(x) counts the number of x in the string
index(x) returns the location of the first occurrence of x
isalpha() returns **True** if every character of the string is a letter

```
s='abcdefghij'
for c in s:
    print(c)
p=s.upper()
print(p)
m=p.replace('A','L')
print(m)
print(s.count('a'))
```

```
s='abcdefghij'
for i in range(len(s)):
    print(s[i])
```

```
s='abcdefghij'
for i in range(len(s)):
    if s[i]=='h':
        print(i)
```

```
Printing name in funny way:
name = input('Enter your name: ')
for i in range(len(name)):
print(name[:i+1])
```

```
Secrete message:
alphabet = 'abcdefghijklmnopqrstuvwxyz'
key =  'uznlwebghjqdyvtkfxompciasr'
secret_message = input('Enter your message: ')
secret_message = secret_message.lower()
for c in secret_message:
    if c.isalpha():
      print(key[alphabet.index(c)],end='')
    else:
      print(c, end='')
```

Lecture V

List

```
L=[1,2,3,4,5]                                  # example of list
L=[]                                            #empty list
L = eval(input('Enter a list: '))    # taking input list
print('The first element is ', L[2])
```

Expression	Result
[7,8]+[3,4,5]	[7,8,3,4,5]
[7,8]*3	[7,8,7,8,7,8]
[0]*5	[0,0,0,0,0]

List function and methods

Function	Description
`len`	returns the number of items in the list
`sum`	returns the sum of the items in the list
`min`	returns the minimum of the items in the list
`max`	returns the maximum of the items in the list

Method	Description
`append(x)`	adds x to the end of the list
`sort()`	sorts the list
`count(x)`	returns the number of times x occurs in the list
`index(x)`	returns the location of the first occurrence of x
`reverse()`	reverses the list
`remove(x)`	removes first occurrence of x from the list
`pop(p)`	removes the item at index p and returns its value
`insert(p,x)`	inserts x at index p of the list

wrong	*right*
`s.replace('X','x')`	`s = s.replace('X','x')`
`L = L.sort()`	`L.sort()`

Assume L=[6, 7, 8]

Operation	New L	Description
`L[1]=9`	`[6,9, 8]`	replace item at index 1 with 9
`L.insert(1,9)`	`[6,9, 7, 8]`	insert a 9 at index 1 without replacing
`del L[1]`	`[6,8]`	delete second item
`del L[:2]`	`[8]`	delete first two items

```
ls=eval(input('enter the list:')) # importing list
print('The list is:',ls)
from random import randint
l=[]                              # new list
count=0
for i in range(50):
    l.append(randint(1,100))
    if l[i]>50:
      count=count+1
print(l,'\n',count)
```

More with List

Function	Description
`choice(L)`	picks a random item from `L`
`sample(L, n)`	picks a group of `n` random items from `L`
`shuffle(L)`	Shuffles the items of `L`

```
from random import shuffle, choice, sample
names = ['Joe', 'Bob', 'Sue', 'Sally', 'Santosh']
print(sample(names, 2))
print(choice(names))
```

join

The join method is in some sense the opposite of split. It is a string method that takes a list of strings and joins them together into a single string. Here are some examples, using the list

L = ['A','B','C']

Operation	Result
`' '.join(L)`	`A B C`
`''.join(L)`	`ABC`
`', '.join(L)`	`A, B, C`
`'***'.join(L)`	`A***B***C`

```
L=['a','b','c']
print('  9999'.join(L))
```

Two dimensional List

```
L=[[]]
```

```
L=[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
print(L[2][3])
```

```
from pprint import pprint
L=[[1,2,3,4],[5,6,7,8],[9,10,11,12]]
#########################################
###### printing the list
for r in range(3):
    for c in range(4):
      print(L[r][c], end=" ")
    print()
#########################################
pprint(L)
##### printing row
print(L[1])
######## printing length and column
print(len(L))
print([L[i][3] for i in range (len(L))])
```

str, int, float

- Convert float, int to string or int into float and vice-versa

Statement	Result
`str(37)`	`'37'`
`str(3.14)`	`'3.14'`
`str([1,2,3])`	`'[1,2,3]'`

Statement	Result
`int('37')`	37
`float('3.14')`	3.14
`int(3.14)`	3

Formatting

- For left justify : >
- For right justify : <
- For center justify : ^
- For integer use: d
- For float use: f
- For string use: s

```
## for integer for left justify
print('{:<3d}'.format(2))
print('{:<3d}'.format(25))
print('{:<3d}'.format(138))
## for integer for right justify
print('{:>3d}'.format(2))
print('{:>3d}'.format(25))
print('{:>3d}'.format(138))
## for integer for center justify
print('{:^5d}'.format(2))
print('{:^5d}'.format(252))
print('{:^5d}'.format(13856))
print('{:^7.2f}'.format(13856))
```

Dictionaries

- A dictionary is a more general version of a list.
- Example: list of days in the months of a year

days = [31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31] # this list

- Here is the dictionary

```
days = {'January':31, 'February':28, 'March':31,
'April':30,'May':31, 'June':30, 'July':31,
'August':31,'September':30, 'October':31,
'November':30, 'December':31}
```

- Use {} for dictionary
- 'January', 'February' etc. are the keys

Changing the value of Key, adding new key and Deleting key

```
days['January']=35    # changing the value of key
days['King']=31       # adding  new key and value
del days['May']             # deleting a key
```

Example: Dictionary

```
Animal = {'dog' : 'has a tail and goes woof!',
'cat' : 'says meow',
'mouse' : 'chased by cats' ,  'lion' :  'King of
Jungle' }

word = input('Enter a word: ')
print('The definition is:', Animal[word])
```

```
alphabet = {'A':100, 'B':200, 'c':300, 'd':400}
letter = input('Enter a letter: ')
if letter in alphabet:
    print('The value is', alphabet[letter])
else:
    print('Not in dictionary')
```

Another of creating dictionary

- **dict** function is another way to create a dictionary. one use for it is kind of like the opposite of the items method:

```
d = dict([('A',100),('B',300)])
```

Function

- Functions are useful for breaking up a large program to make it easier to read and maintain.
- Also useful if find yourself writing the same code at several different points in your program.
- Functions are defined with the **def** statement. The statement ends with a colon, and the code that is part of the function is indented below the **def** statement.

```
def print_hello(n):
    print('Hello! '*n)
print_hello(5)
```

```
def convert(t):
    return t*9/5+32
print(convert(20))
```

```
# defining the factorial
def fact(x):
    s=1
    for i in range(1,x+1):
        s=s*i
    return s
print(fact(5))
```

```
def draw_square():
    print('*' * 15)
    print('*', ' '*11, '*')
    print('*', ' '*11, '*')
    print('*' * 15)
draw_square()
```

```
from math import pi, sin
def deg_sin(x):
    return sin(pi*x/180)
print(deg_sin(30))
```

Nested function

```
def f(x):
    def f1(x):
        s=x**x
        return s
    def f2(x):
        y=x**3
        return y
    g=f1(x)+f2(x)
    return g
print(f(2))
```