UNIT 1

Reference textbook :

An introduction to Python programming

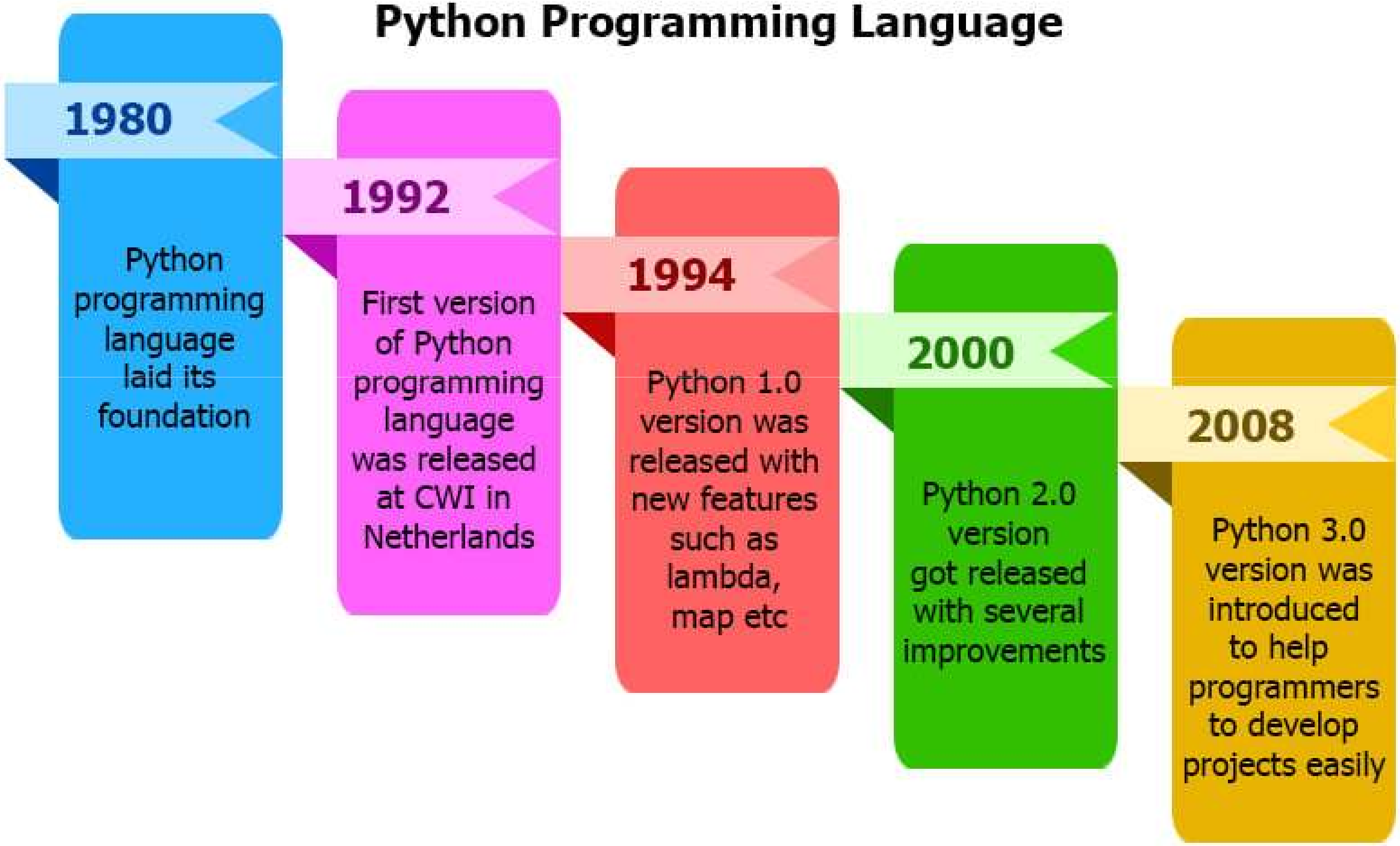
Author: Gowrishankar S

History of Python

- Python was introduced in late 80’s (around 1990) by **Guido von Rossum**.
- It was based on a programming language ABC
- Python’s first release **Python 1.0** happened between **1992 and upto 1994**.
- Python **2.0** has arrived in the **year 2000** which brought in additions incorporating number of important improvements & marked shift in evolutionary path of the language.
- **Python 3.0** was released at the **end of 2008** which cleaned up many of the inconsistencies in the design of various releases of Python 2. However, it’s not backward compatible.
- **Latest stable version of python is 3.11.3**

Guido von rossum

A Brief History Of Development of Python Programming Language



1980

Python programming language laid its foundation

1992

First version of Python programming language was released at CWI in Netherlands

1994

Python 1.0 version was released with new features such as lambda, map etc

2000

Python 2.0 version got released with several improvements

2008

Python 3.0 version was introduced to help programmers to develop projects easily

History of Python: Story behind the name

- Often people assume that the name Python was written after a snake.
- Even the logo of Python programming language depicts the picture of two snakes, blue and yellow.
- Back in the 1970s, there was a popular BBC comedy tv show called Monty Python’s Fly Circus and Van Rossum happened to be the big fan of that show. So when Python was developed, Rossum named the project ‘Python’.

Applications of Python

- **Web and Internet Development**
  - Frameworks such as Django and Pyramid.
  - Micro-frameworks such as Flask and Bottle.
  - Advanced content management systems such as Plone and django CMS.
- **Python's standard library supports many Internet protocols:**
  - HTML and XML
  - JSON
  - E-mail processing.
  - Support for FTP, IMAP, and other Internet protocols.
  - Easy-to-use socket interface.

- **Scientific , Numeric and data science**
  - Python is widely used in scientific and numeric computing:
  - **SciPy** is a collection of packages for mathematics, science, and engineering.
  - **Pandas** is a data analysis (data science) and modeling library.
  - **IPython** is a powerful interactive shell that features easy editing and recording of a work session, and supports visualizations and parallel computing.
  - The **Software Carpentry Course** teaches basic skills for scientific computing, running bootcamps and providing open-access teaching materials.

- **Machine Learning and Artificial Intelligence**
  - Python along with its inbuilt libraries and tools facilitate the development of AI and ML algorithms.
  - Some of the inbuilt libraries and tools that enhance AI and ML processes are:
    - **Numpy** for complex data analysis
    - **Keras** for Machine learning
    - **Seaborn** for data visualization

- **Desktop GUIs**
  - The **Tk GUI library** is included with most binary distributions of Python.
  - Some toolkits that are usable on several platforms are available separately:
    - **wxWidgets**
    - **Kivy**, for writing multitouch applications.
    - Qt via pyqt or pyside
  - Platform-specific toolkits are also available:
    - GTK+
    - Microsoft Foundation Classes through the win32 extensions

- **Software Development**
  - Python is often used as a support language for software developers, for build control and management, testing, and in many other ways.
  - This language offers amazing features like:
    - Platform independence
    - Inbuilt libraries and frameworks to provide ease of development.
    - Enhanced code reusability and readability
    - High compatibility
  - SCons for build control.
  - Buildbot and Apache Gump for automated continuous compilation and testing.
  - Roundup or Trac for bug tracking and project management.

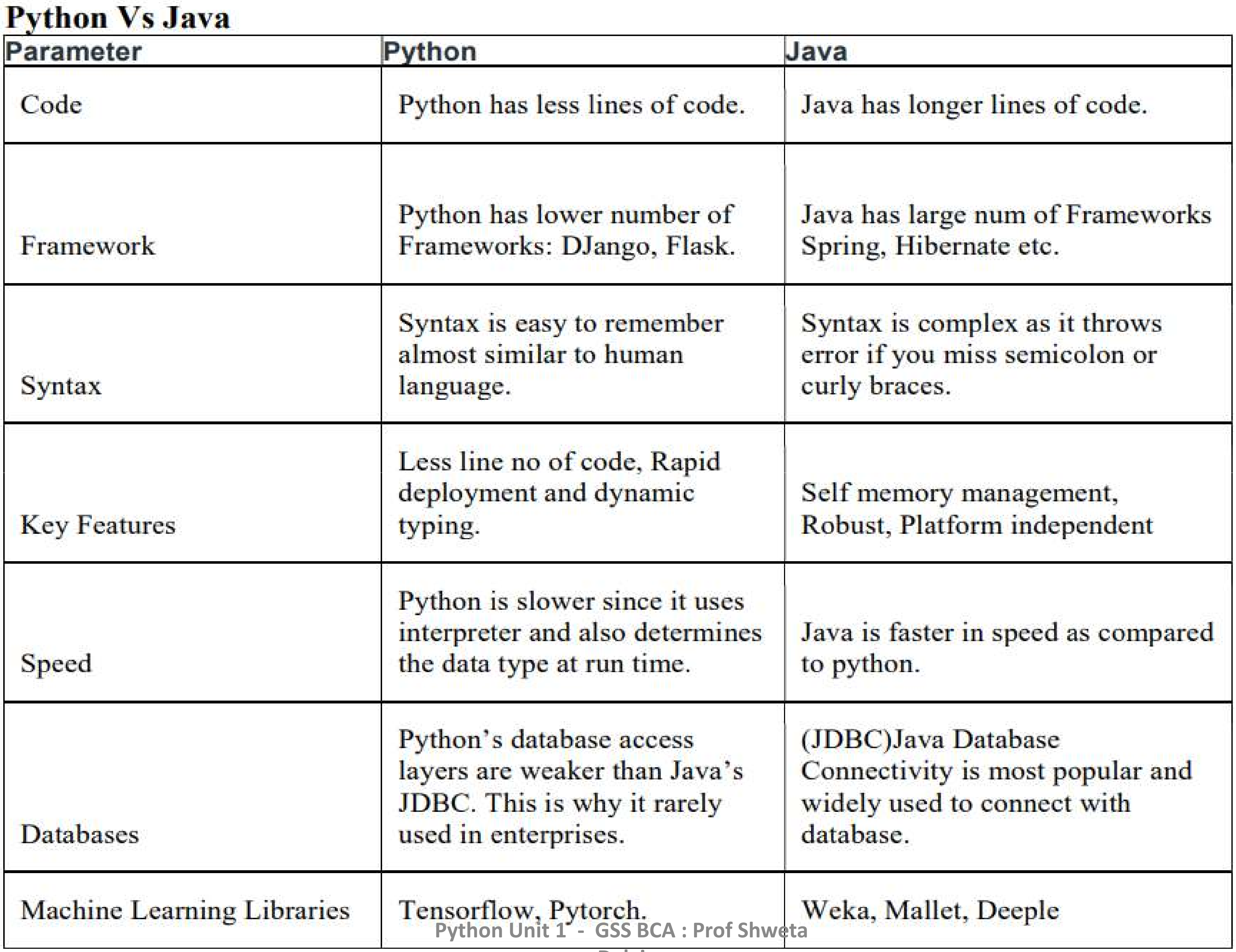
- **Business Applications**
  - Python is also used to build ERP and e-commerce systems:
  - Odoo is an all-in-one management software that offers a range of business applications that form a complete suite of enterprise management applications.
  - Tryton is a three-tier high-level general purpose application platform.

- **Game Development**
- Python has proved to be an exceptional option for game development.
- The presence of popular 2D and 3D gaming libraries like **pygame, panda3D, and Cocos2D** make the game development process completely effortless.

- **Audio and Visual Applications**
  - Python is equipped with a lot of tools and libraries to accomplish your task flawlessly.
  - Applications that are coded in Python include popular ones like Netflix, Spotify, and YouTube.
  - This can be handled by libraries like **Dejavu, Pyo, Mingus, SciPy and OpenCV**

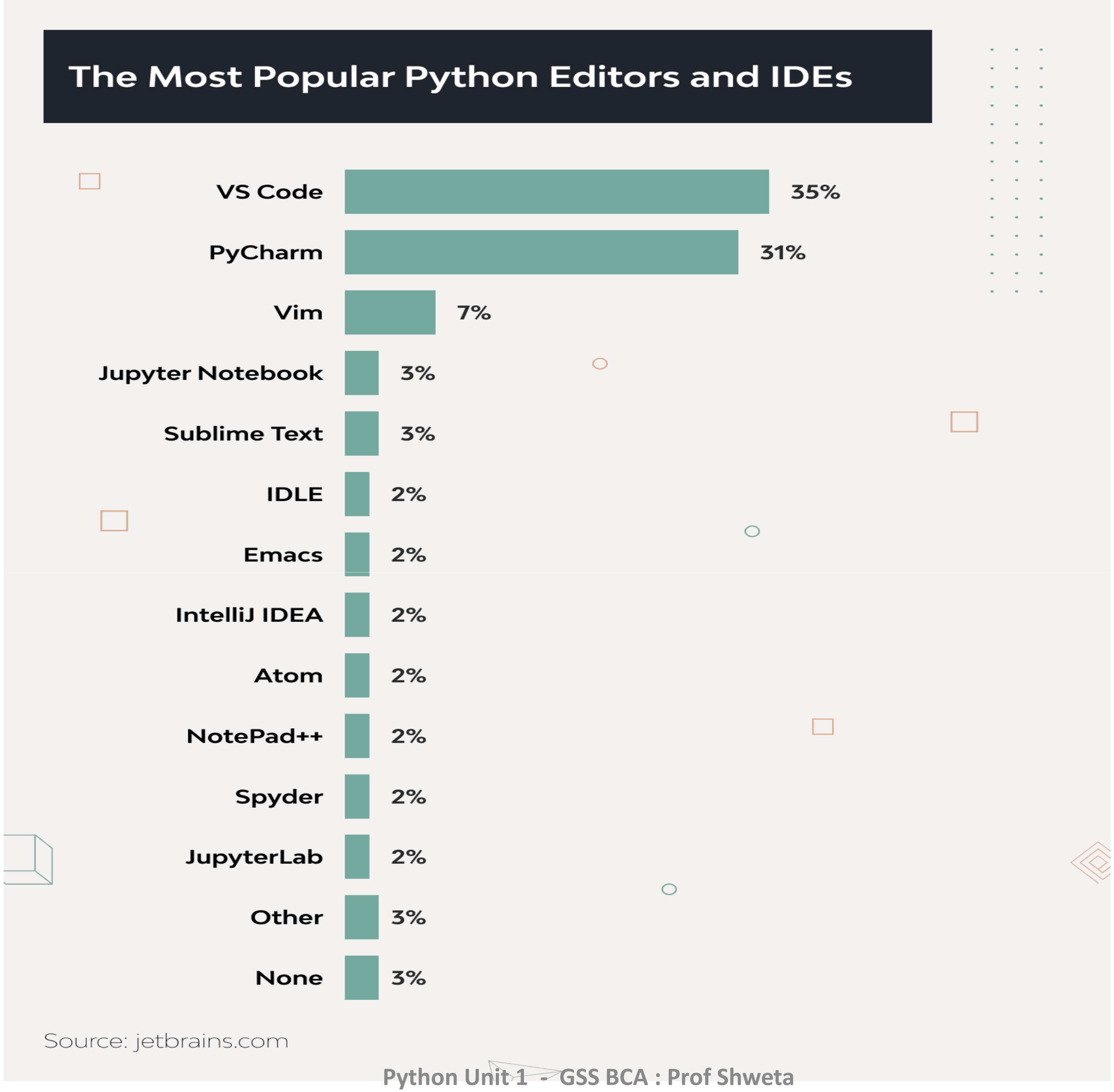
- **CAD (computer-aided design) Applications**
  - it is the process of creating 3D and 2D models digitally and is used by architects, product designers and construction managers to design products with extremely high consistency.
  - Python can be embedded with amazing applications like Blender, FreeCAD, open cascade etc.
  - These provide enhanced features like technical drawing, dynamic system development, recordings, file export, and import.

- **Web Scraping Application**
  - Web scraping is an automated process used to extract information from websites in an easier and faster way. The information is used by researchers, organizations, and analysts for a wide variety of tasks.
  - Python has a wide range of features that make it suitable for web scraping
  - A wide range of libraries and tools like pandas, matplotlib, and Selenium makes the web scraping process easy and efficient.
  - Easy to use and understand



Python Vs Java

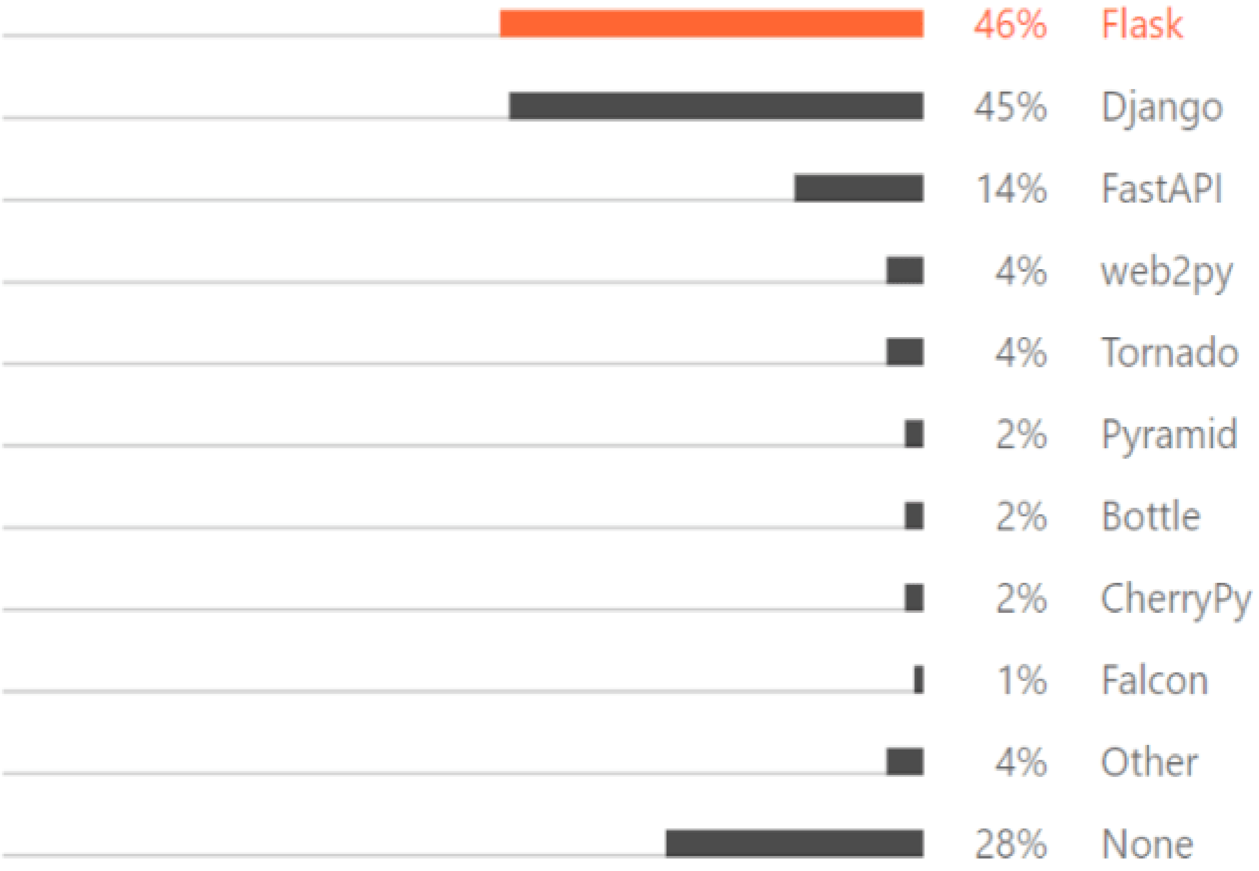
Parameter	Python	Java
Code	Python has less lines of code.	Java has longer lines of code.
Framework	Python has lower number of Frameworks: DJango, Flask.	Java has large num of Frameworks Spring, Hibernate etc.
Syntax	Syntax is easy to remember almost similar to human language.	Syntax is complex as it throws error if you miss semicolon or curly braces.
Key Features	Less line no of code, Rapid deployment and dynamic typing.	Self memory management, Robust, Platform independent
Speed	Python is slower since it uses interpreter and also determines the data type at run time.	Java is faster in speed as compared to python.
Databases	Python's database access layers are weaker than Java's JDBC. This is why it rarely used in enterprises.	(JDBC)Java Database Connectivity is most popular and widely used to connect with database.
Machine Learning Libraries	Tensorflow, Pytorch.	Weka, Mallet, Deeple



The Most Popular Python Editors and IDEs

Source: jetbrains.com

What web frameworks / libraries do you use in addition to Python?



- **Python IDLE**
  - Python IDLE comes with python package.
  - IDLE stands for Integrated Development and Learning Environment
  - ANACODA and Canopy are the popular IDE's of Python.

Python IDLE's features-

- text editor
- shell with syntax highlighting
- integrated debugger
- File menu which has options of : new file, open file, save file etc
- Edit menu which has options of: copy, paste, find, indent, comment etc

Basic Elements of Python:

- **Python program (aka script)** is sequence of definition and commands
- These definitions are evaluated and commands are executed by python interpreter in something called as **shell**.
- A **command**, often called statement instructs the interpreter to do something.
- Example – print(“Hello World”) will instruct the interpreter to call the function print(), which will print Hello World to the window associated with shell.

- Python is a high level interpreted language.
- There are two ways to use the interpreter:

  a) command line mode

  b) script mode
- In command-line mode, you type Python programs and the interpreter prints the result
- In script mode you write a program in a file called script and use the interpreter to execute the contents of the file.

Python command line mode

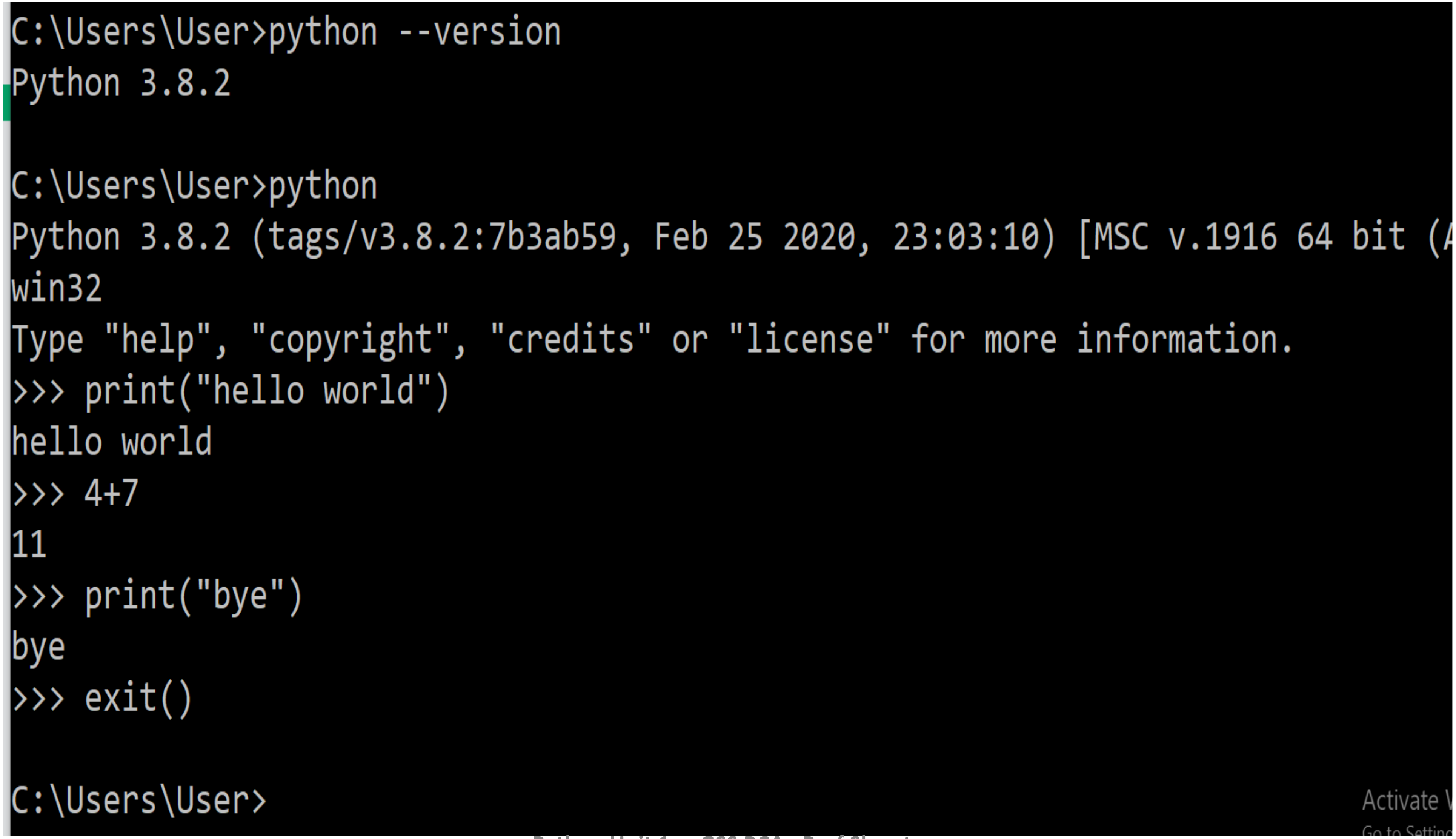
- Python code can be run as a command line itself.
- To do this, just open command line by typing cmd in search tab of your computer .
- Now to check if python is installed, at the command line type,

```
python --version
```

If python is installed you will get the version as output

- Now to start the python interpreter type Python or py at the command line. You will get the python prompt >>>
- Then start typing your python code.

Example on python command line

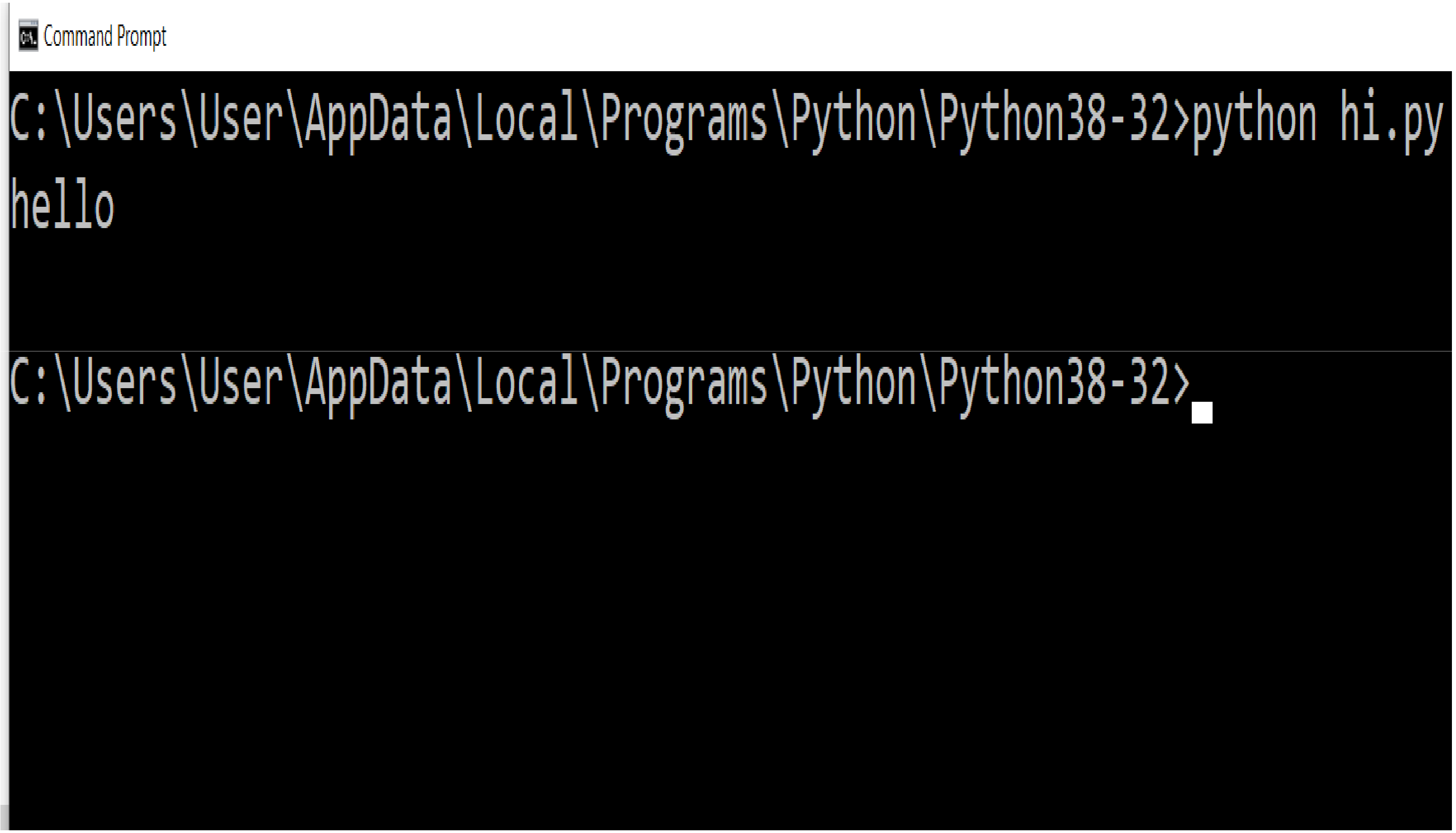


```
C:\Users\User>python --version
Python 3.8.2

C:\Users\User>python
Python 3.8.2 (tags/v3.8.2:7b3ab59, Feb 25 2020, 23:03:10) [MSC v.1916 64 bit (A
win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("hello world")
hello world
>>> 4+7
11
>>> print("bye")
bye
>>> exit()

C:\Users\User>
```

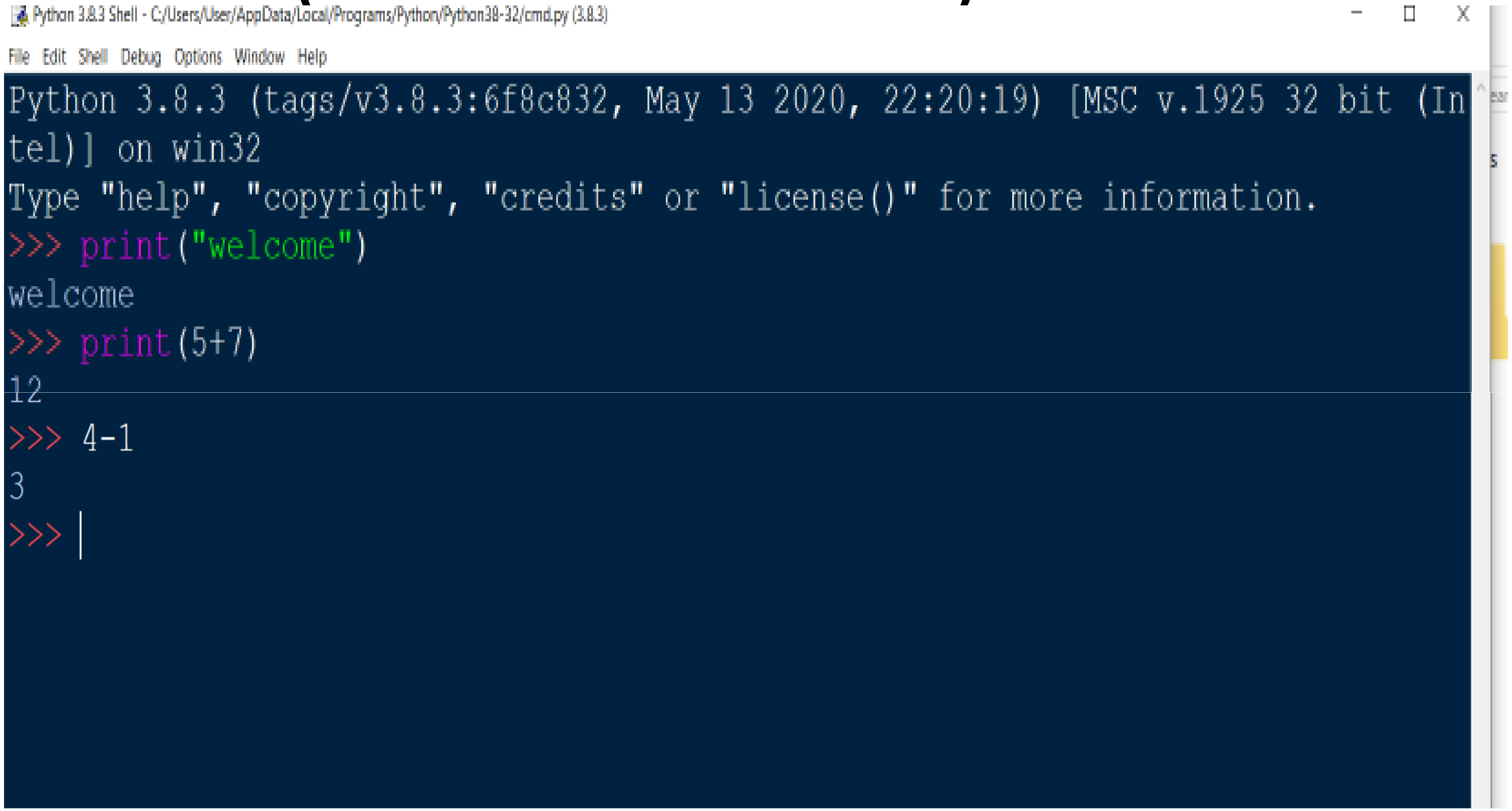
Executing a python file hi.py in command line



```
C:\Users\User\AppData\Local\Programs\Python\Python38-32>python hi.py
hello

C:\Users\User\AppData\Local\Programs\Python\Python38-32>
```

Executing in interactive mode (command mode) in IDLE



Python 3.8.3 Shell - C:/Users/User/AppData/Local/Programs/Python/Python38-32/cmd.py (3.8.3)
File Edit Shell Debug Options Window Help

```
Python 3.8.3 (tags/v3.8.3:6f8c832, May 13 2020, 22:20:19) [MSC v.1925 32 bit (In
tel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("welcome")
welcome
>>> print(5+7)
12
>>> 4-1
3
>>> 
```

Executing in IDE
Here we are using IDLE IDE

- open an IDE . Example : IDLE , pycharm etc

steps for executing in IDLE:

1. open IDLE
2. click on file menu → new file
3. type the program
4. click on file menu → save
5. click on run menu → run module

Comments in python

- To enhance readability of code comments are added.
- Comments are statements which are not interpreted by python.
- # is used for single line comment.

  #this is a single line comment
- ”””” (3 double quotes) or “’’ (3 single quotes) are used for multiline comments.
- “””this is a multiline comment

  so we can write on multiple lines “””

Simple python program

```
# welcome
# this is commenting lines
print("welcome");
print(5);
print("apples", 10*40);
#wel
'''
this is a multine comment
i cannot see this
'''

"""
this is also a multline comment
wow is this possible
"""
```

Objects

- Objects are the core thing that Python program can manipulate.
- Objects are of two types:

A) **Scalar objects** - are indivisible and do not have internal structure

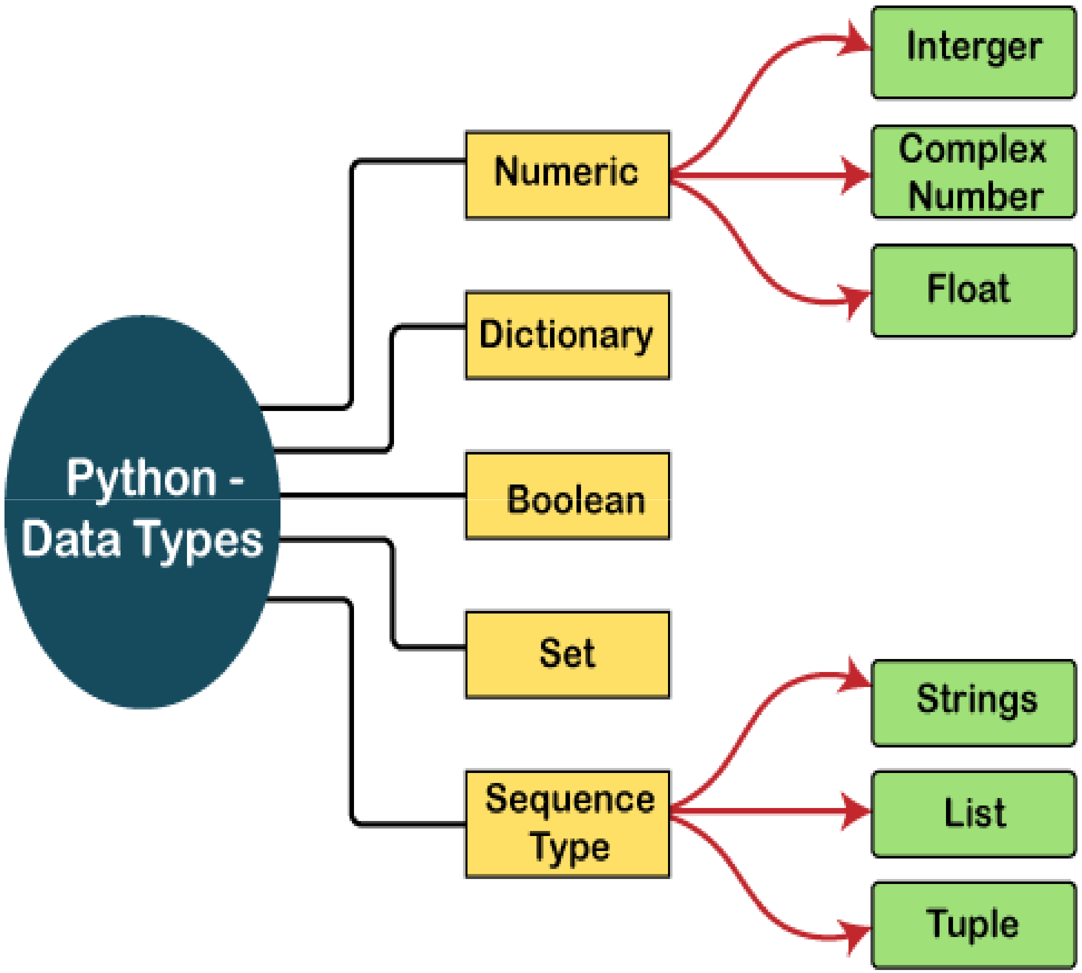
Ex: atoms of language (int, float, bool etc)

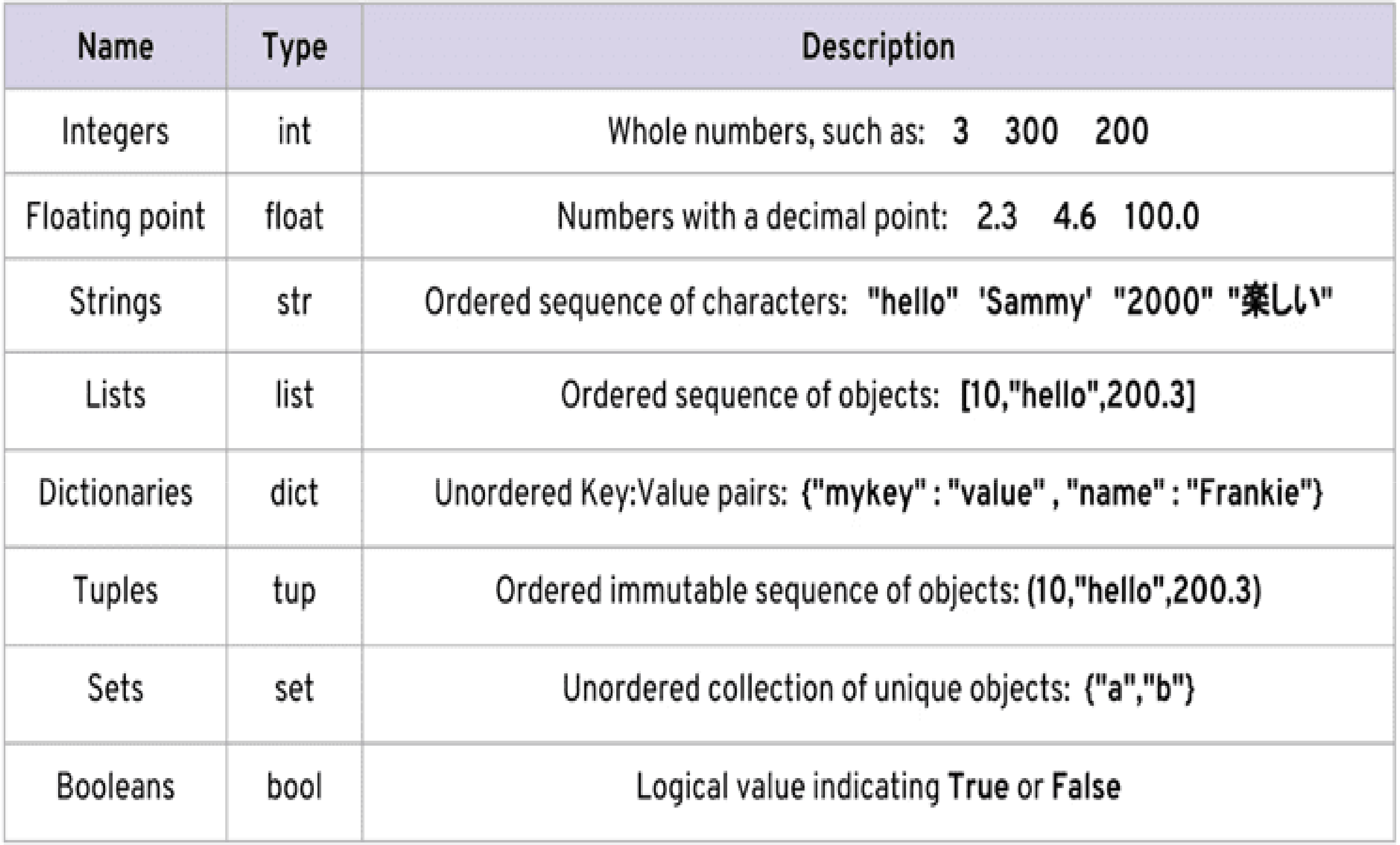
Python has four types of Scalar objects

1. int which is used to represent integer. Example (-3. 5, 11200) etc
2. float is used to represent real numbers.
   Literals of float are written like (3.0, 3.17, -28.72)
3. bool is used to represent Boolean values. True or False
4. None is type with single value.

B) **Non scalar objects** - have internal structure.

(Ex: strings, list, tuple, dictionary)


Python - Data Types
Numeric
Interger
Complex Number
Float
Dictionary
Boolean
Set
Sequence Type
Strings
List
Tuple



Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

type() function

- to find the datatype of a variable in python
  example

```
>>> type(2)
        <class 'int'>
>>> type(3.2)
        <class 'float'>
>>> type("hi")
        <class 'str'>
```

Variables

- Variable is name/identifier that refers to a value

  Example- pi = 3.142, it first binds variable name pi to objects of type float.
- In python, a variable is just a name.
- Assignment statement associates the name to the left of
- the = symbol with object denoted by expression to the right of =.
- Apt choice of variable names plays an important role in enhancing readability of code.

  Example a = 3.142 pi = 3.142
- Here, we read the statements- variable a is not giving much clarity but variable pi suggest something related to circle. Such apt names are called mnemonic variables

Rules for declaring Variable –

- In python, variable names can be **arbitrary long**.
- Variables can **contain both letters and numbers**. (It **cannot start with a number**)
- Variable names are **case sensitive** (a and A are two different variable names)
- Special character **_(underscore) is allowed**. (It is used when we have multiple words)
- Python **keywords(reserve words) cannot** be used as variable names.

Statement

- Statement - is a piece of code that python interpreter can execute.
- We have seen two kinds of statements: print and assignment.
- The result of a print statement is a value. Assignment statements don't produce a result.
- A script(python program) usually contains a sequence of statements.

Example – Assignment statement x = 2.

Expression

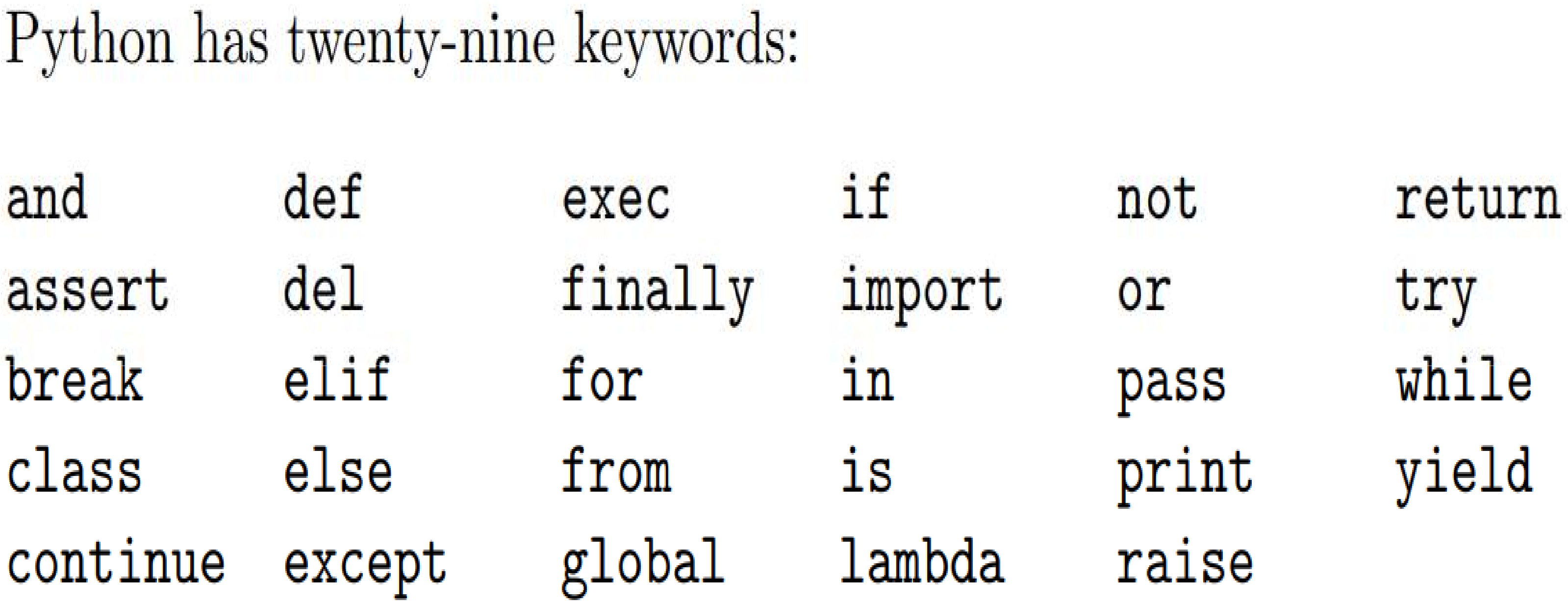
- Expression- is combination of values, variables and operators.
- Operators are special symbols that represent computations like addition, subtraction etc.

Example :

a = 25, b = 10, c = a + b

Here a, b, c are operands (variables). +, = are operators, 25, 10 are values.

Keywords in python



Python has twenty-nine keywords:

and
assert
break
class
continue
def
del
elif
else
except
exec
finally
for
from
global
if
import
in
is
lambda
not
or
pass
print
raise
return
try
while
yield

Console input and output

- Console (also known as Shell) is a command line interpreter that processes input from the user, or one command at a time.
- If there are no errors, the command is executed and the necessary output is produced; otherwise, an error message is displayed.
- The three greater than symbols >>> serve as the Python console's default prompt.

- **The print() function** prints the specified message to the screen, or other standard output device.
- The message can be a string, or any other object, the object will be converted into a string before written to the screen.
- Syntax

  print(object(s), sep=separator, end=end, file=file, flush=flush)

example:

```
print("Hello", "how are you?")
print("Hello", "how are you?", sep="---")
 x = ("apple", "banana", "cherry")
print(x)
```

Accepting user Input

- Python3 has input() function that can be used to get input from a user. It takes string as argument and displays it as prompt in the shell.
- It waits for user to type something, followed by hitting enter.
- Typed line is treated as string and becomes the value returned by the function.

  Ex: name = input(“Enter your name”)

- As input function takes input in string format, there is need for type conversion (type casts)

- Python3 has input() function similar to raw_input() of Python2.

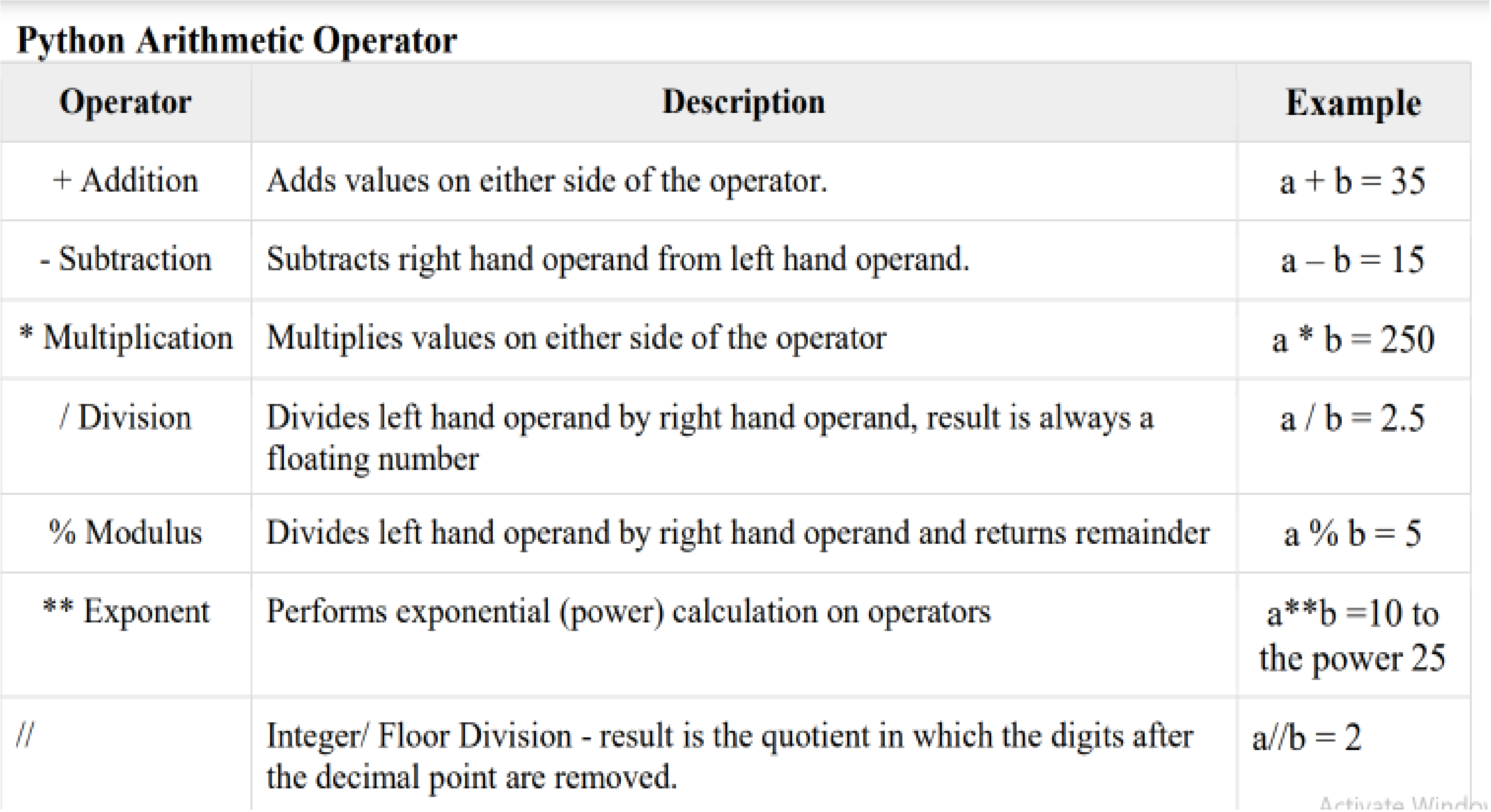
Type casting the input

- Python's built-in input() method always produces an object of the str(string) class
- There may be situations where we need integer input from the user or console.
- Hence we need to type convert the input to integer

  Ex: num = int(input("Enter a number"))
- Similarly we can type cast input to float

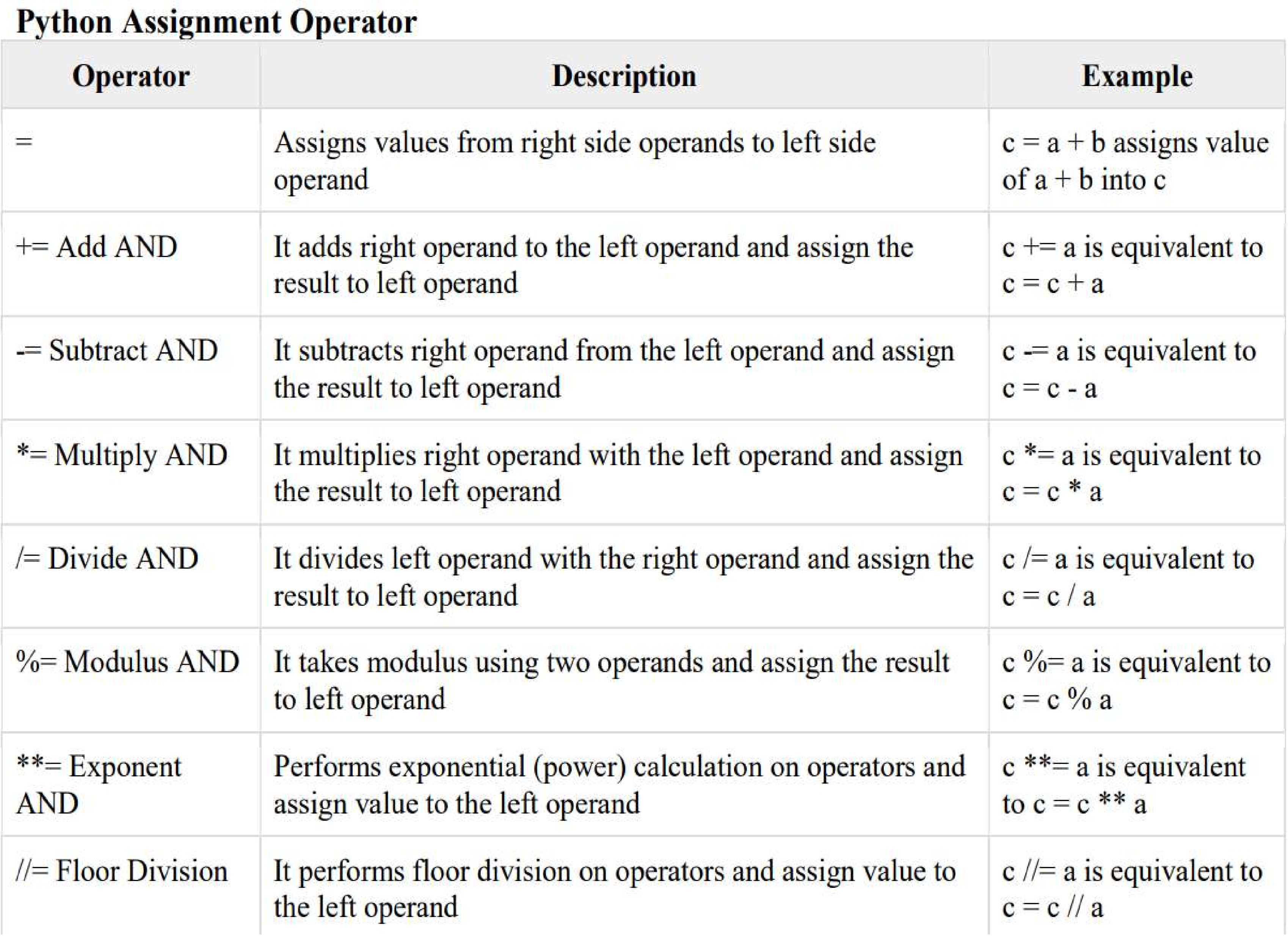
  num1 = float(input())
- Also we can type cast input to string

  S1 = str(input())



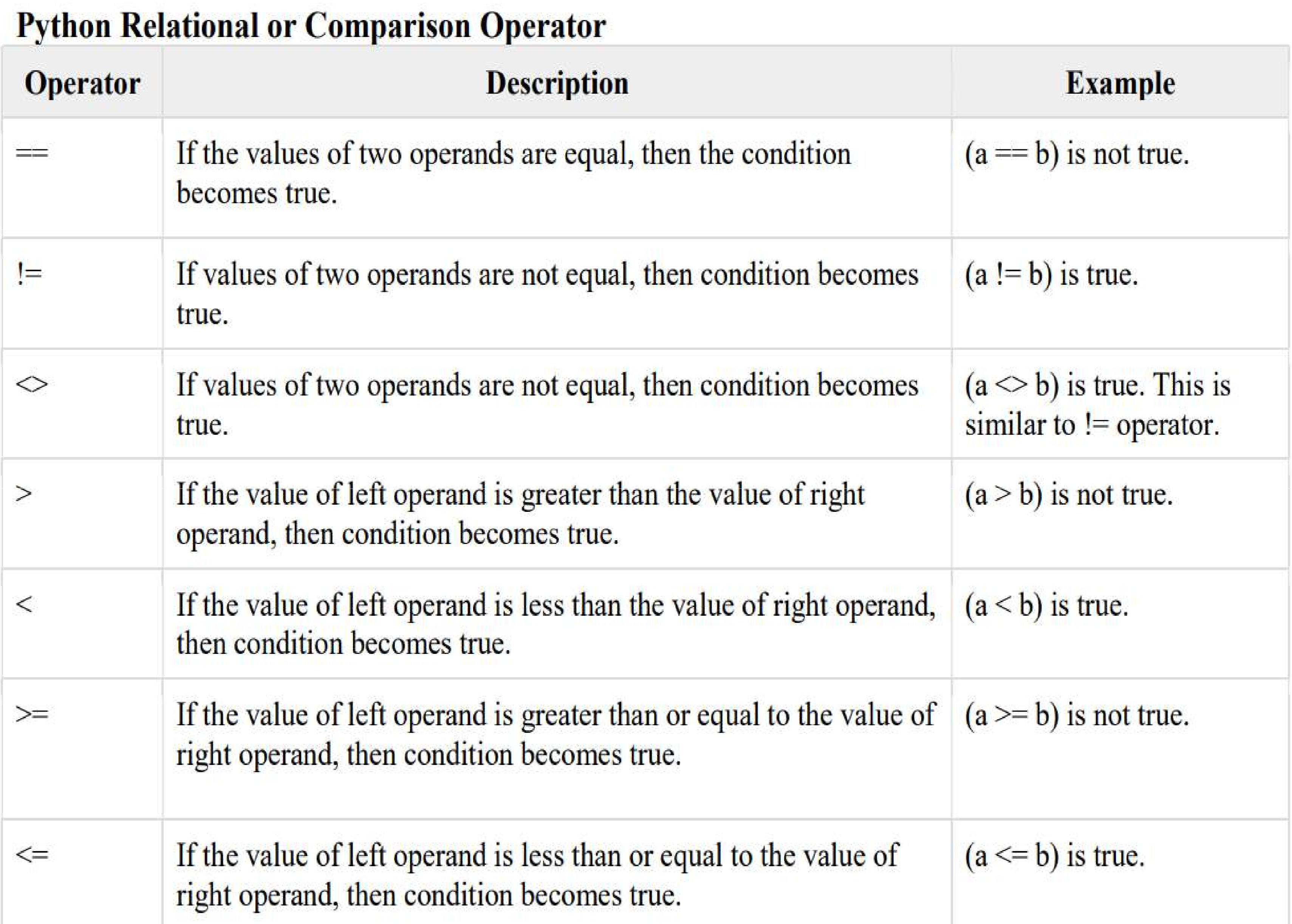
Python Arithmetic Operator

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 35
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = 15
* Multiplication	Multiplies values on either side of the operator	a * b = 250
/ Division	Divides left hand operand by right hand operand, result is always a floating number	a / b = 2.5
% Modulus	Divides left hand operand by right hand operand and returns remainder	a % b = 5
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 25
//	Integer/ Floor Division - result is the quotient in which the digits after the decimal point are removed.	a//b = 2



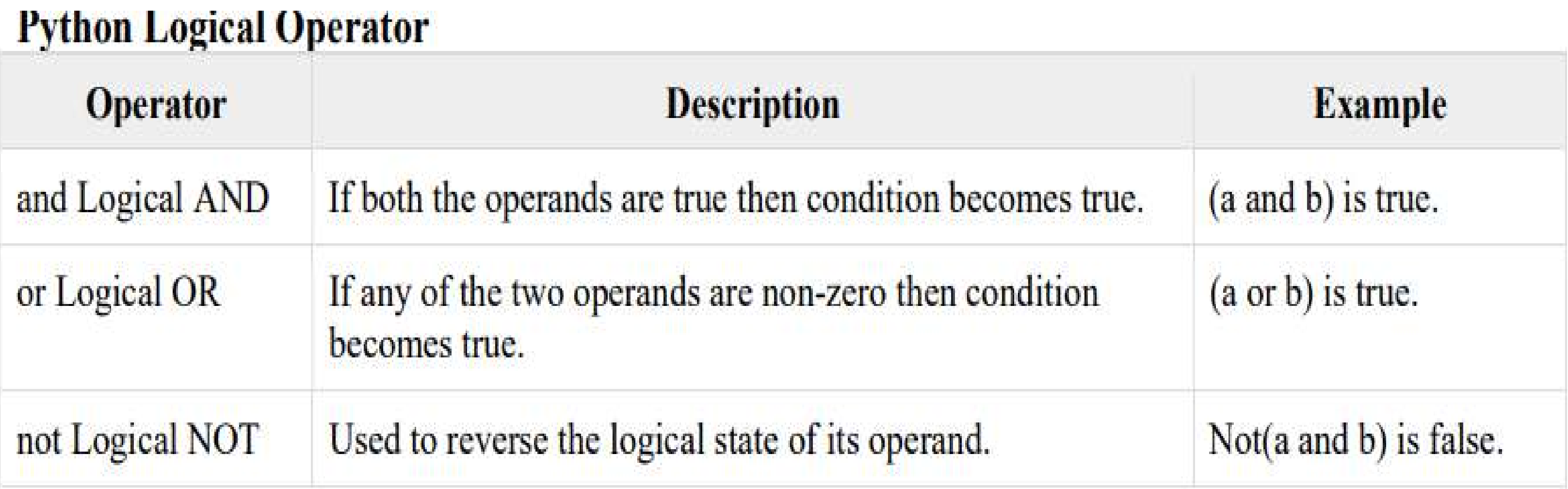
Python Assignment Operator

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
**= Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	c **= a is equivalent to c = c ** a
//= Floor Division	It performs floor division on operators and assign value to the left operand	c //= a is equivalent to c = c // a



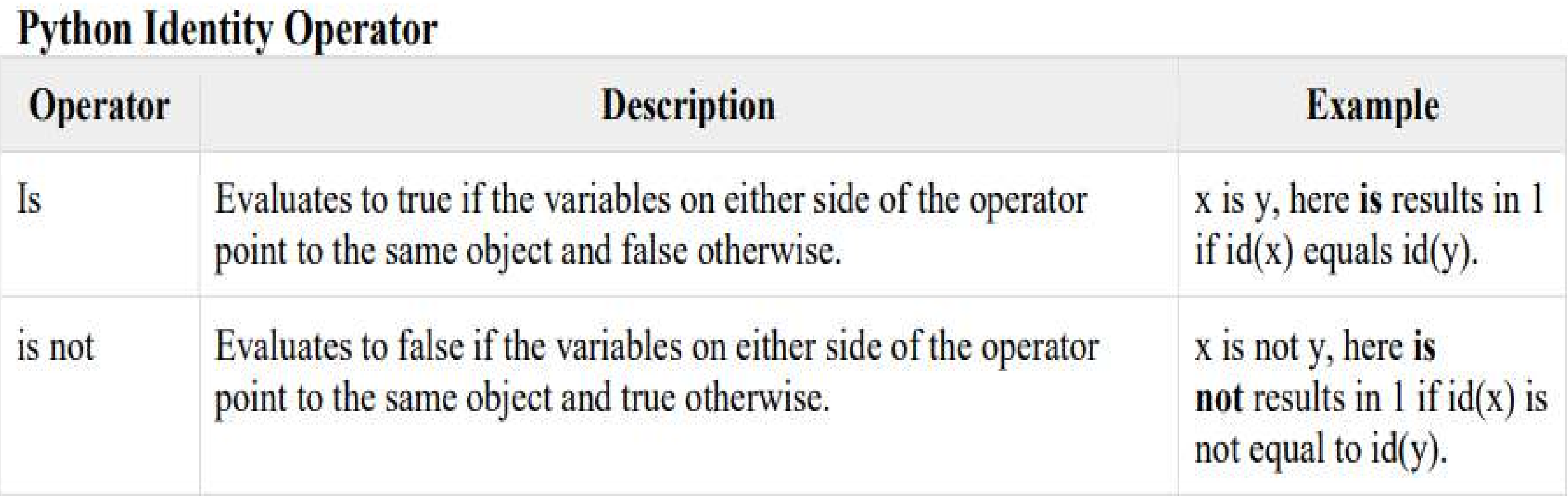
Python Relational or Comparison Operator

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	(a != b) is true.
<>	If values of two operands are not equal, then condition becomes true.	(a <> b) is true. This is similar to != operator.
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.



Python Logical Operator

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.



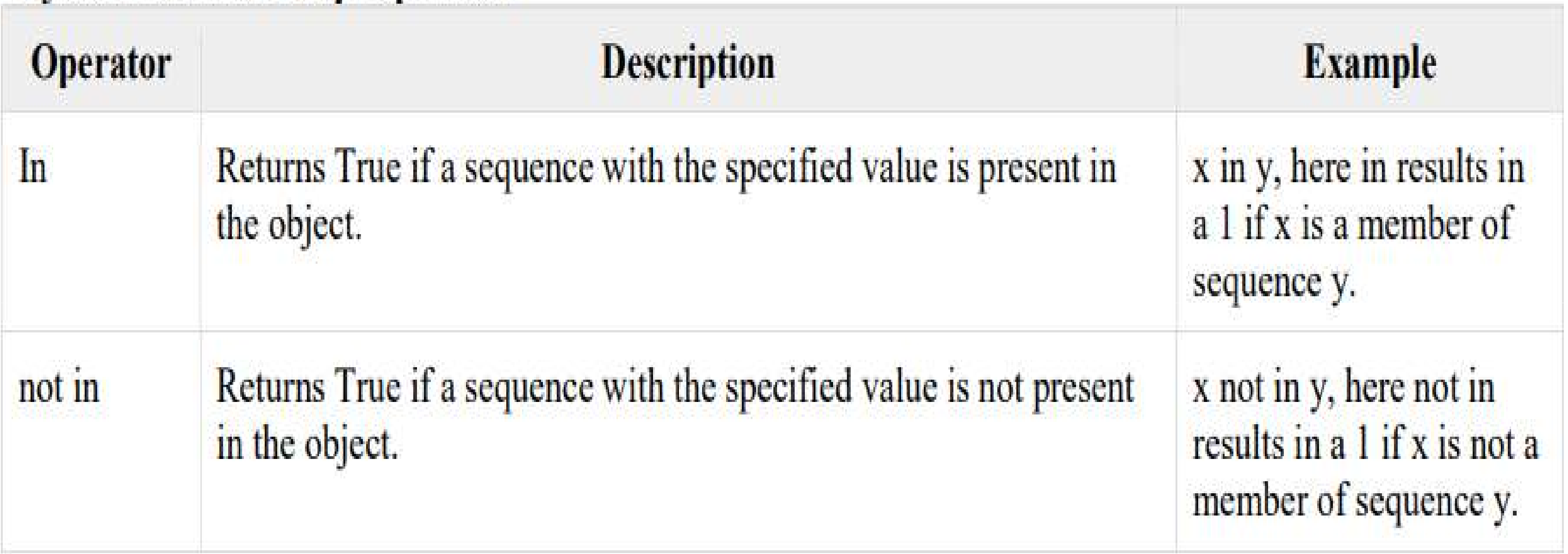
Python Identity Operator

Operator	Description	Example
Is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

```
x = [“steve", “jobs"]     #list 1 defined
y = [“steve", “jobs"]     #list2  defined
z = x
print(x is z)
# returns True because z is the same object as x
print(x is y)
# returns False because x is not the same object as y, even if they have the same content
print(x == y)
# to demonstrate the difference betweeen "is" and "==": this comparison returns True because x is equal to y

print(x is not y)
# returns True because x is not the same object as y, even if they have the same content
```

Python Membership Operator



Operator	Description	Example
In	Returns True if a sequence with the specified value is present in the object.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Returns True if a sequence with the specified value is not present in the object.	x not in y, here not in results in a 1 if x is not a member of sequence y.

```
x = ["steve", "jobs"]    #list defined

print("steve" in x)
# returns True because a sequence with the value "steve" is in the list

print("mark" in x)
# returns False because a sequence with the value "mark" is not in the list
```

Operator Precedence

- When more than one operator appears in an expression , precedence describes the order in which operations are to performed.
- example   cs= d*(a+b)
- Python follows the same precedence rules for its mathematical operators that mathematics does.

  PEMDAS (Parenthesis Exponent Multiplication Division Addition Subtraction)
- Operators with the same precedence are evaluated from left to right

Examples on precedence

- 2 * (3-1) = 4
- (1+1)**(5-2) = 8
- 3*1**3 = 3
- 2*3-1 = 5
- 2/3-1 = -1

Indentation

- Indentation in python is used to identify block of code.
- other programming languages use some sort of bracketing symbols { } to denote blocks
- Rules for indentation
  - The first line of Python code can't have an indentation, it will throw IndentationError.
  - You should avoid mixing tabs and whitespaces to create an indentation.
  - It is preferred to use whitespace than the tab character.
  - The best practice is to use 4 whitespaces for the indentation
  - you have to use the same number of spaces in the same block of code

Example on indentation

```
def foo():
    print("Hi")

    if True:
        print("true")
    else:
        print("false")

print("Done")
```

Libraries in Python

- A library is an initially merged collection of code scripts that can be used iteratively to save time.
- A Python library is a group of interconnected modules. It contains code bundles that can be reused in a variety of programs.
- How a python library works?
  - The library files have a DLL extension in the MS Windows environment (Dynamic Load Libraries). When we import a library to our program and run it, the linker looks for that library automatically. It extracts the library's functions and then interprets the program accordingly.

Top 10 Python Libraries

Pandas
Data analysis and manipulation

NumPy
Mathematical functions

Matplotlib
Data visualisations

SeaBorn
Data visualisations

Tensorflow
Machine Learning

Keras
Deep Learning

SciPy
Scientific computing

PyTorch
Machine Learning

Scrapy
Web crawling

SQLModel
Interact with SQL databases

How to Import or use a python library in our code

1) To import a library just use the keyword import followed by the library name

```
# Importing math library
import math
A = 16
print(math.sqrt(A))
```

2) To import a specific items from a library use the keyword “ from “

```
# Importing specific items
from math import sqrt, sin
A = 16
B = 3.14
print(sqrt(A))
print(sin(B))
```

Random module

- Python has a built-in module that you can use to make random numbers.
- The seed() method is used to initialize the random number generator.
- The random number generator needs a number to start with (a seed value), to be able to generate a random number.

Python control flow

- Python supports a set of control flow statements that will help to conditionally execute blocks of code.
- The control flow statements in Python Programming Language are

1. Sequential Control Flow Statements: This refers to the line by line execution, in which the statements are executed sequentially, in the same order in which they appear in the program.

2. Decision Control Flow Statements: Depending on whether a condition is True or False, the decision structure may skip the execution of an entire block of statements or even execute one block of statements instead of other (if, if…else and if…elif…else).

3. Loop Control Flow Statements: This is a control structure that allows the execution of a block of statements multiple times until a loop termination condition is met (for loop and while loop). Loop Control Flow Statements are also called Repetition statements or Iteration statements.

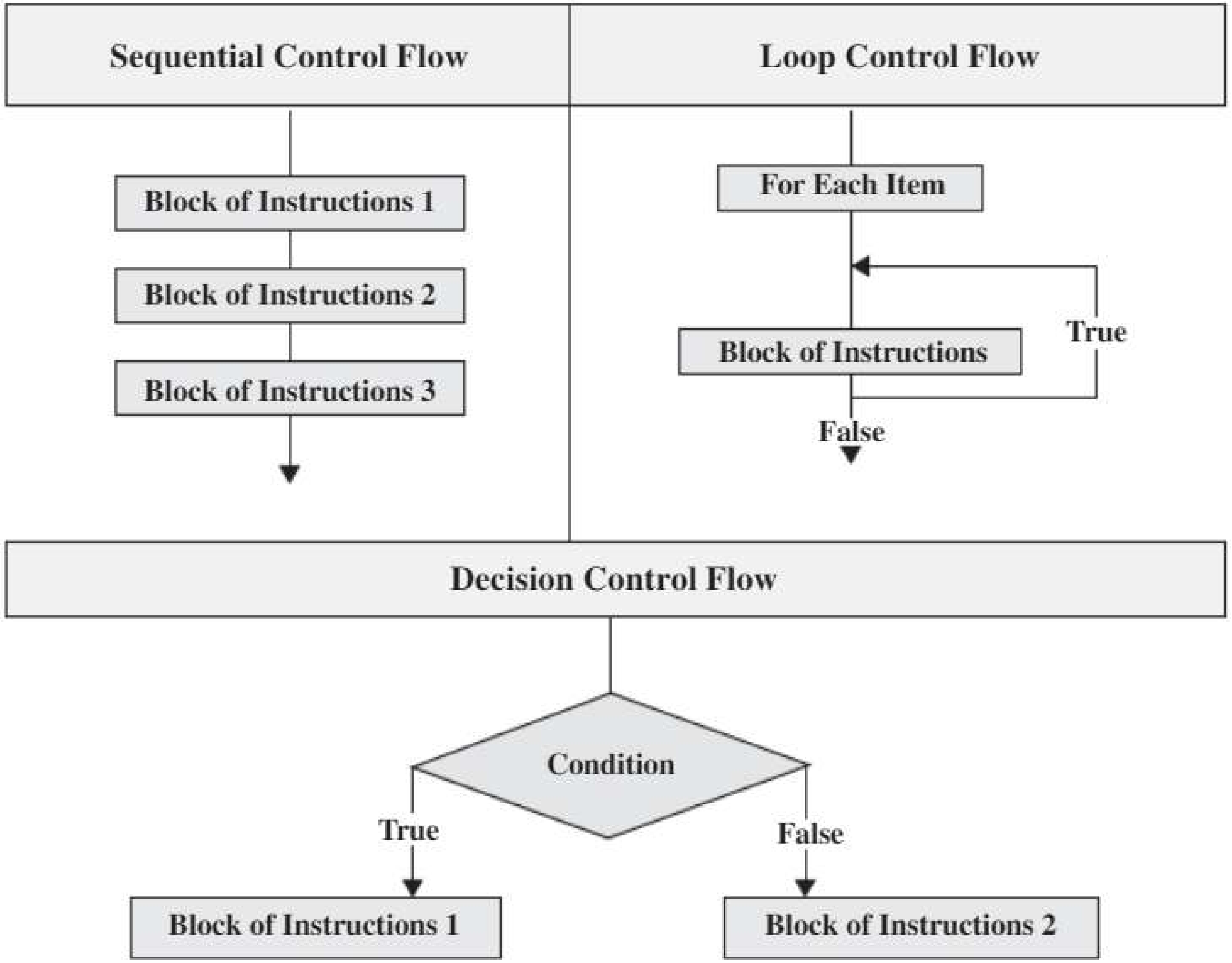


FIGURE 3.1
Forms of control flow statements.

The IF decision control flow (conditional execution):

- The if decision control flow statement starts with if keyword and ends with a colon. The expression in an if statement should be a Boolean expression.
- If the Boolean expression evaluates to True then statements in the if block will be executed; otherwise the result is False then none of the statements are executed.
- In Python, the if block statements are determined through indentation and the first unindented statement marks the end.
- Syntax

```
if <boolean expression> :
        statement(s)
```

- Example

```
if 20 > 10:
        print("20 is greater than 10")
```

If-else decision flow (alternative execution)

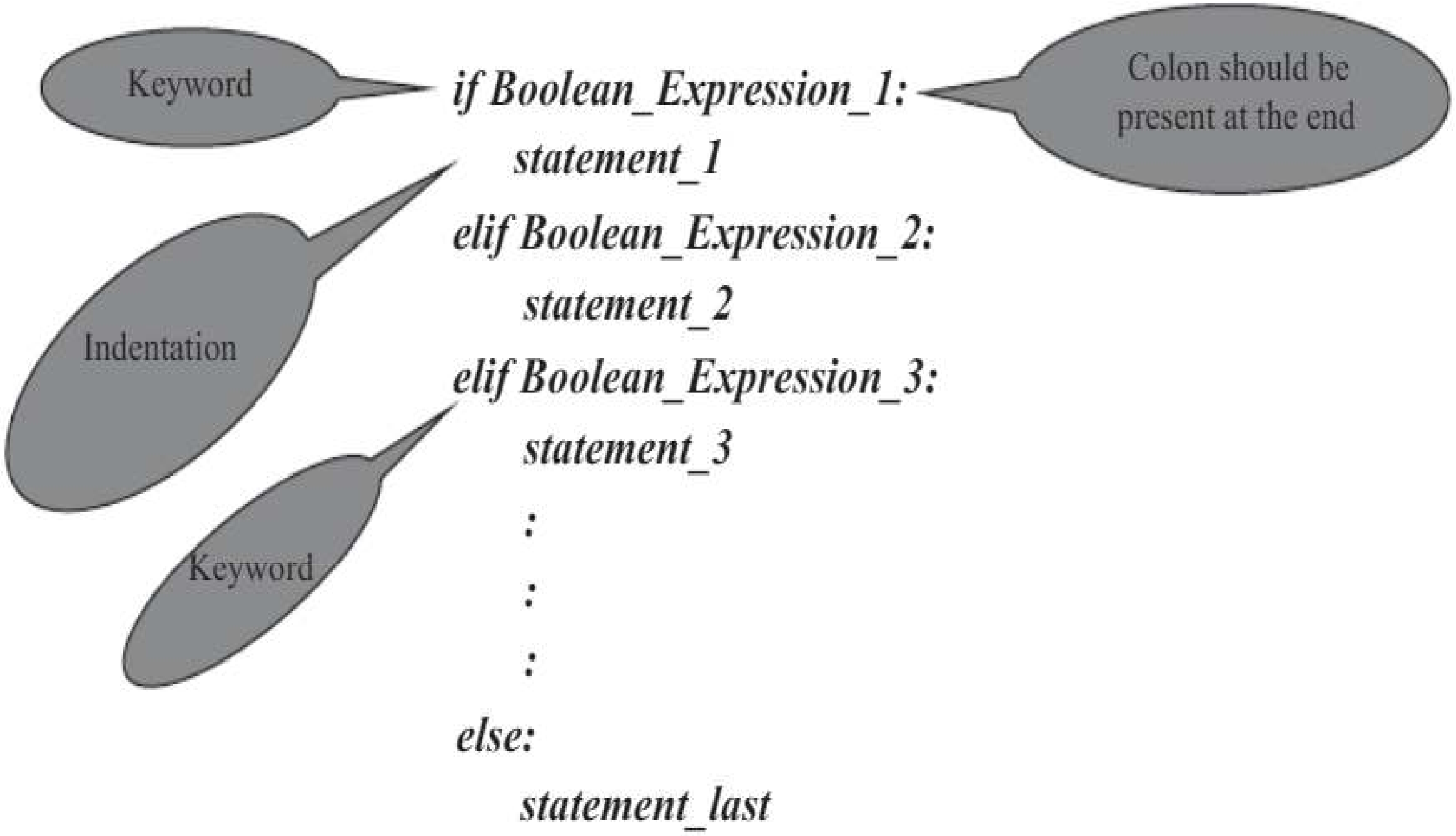
- An if statement can also be followed by an else statement which is optional.
- An else statement does not have any condition.
- Statements in the if block are executed if the Boolean_Expression is True. If the Boolean_Expression is False the else block is executed.

```
if <boolean expression> :
                if block statement(s)
else:
                else block statements(s)
```

```
d = int(input("Enter a number"))
print(“The number is ", d)
if d % 2 == 0:
    print(" It is Even number")
else:
    print(" It is Odd number")
```

The if…elif…else Decision Control Statement (chained conditionals)

- The if…elif…else is also called as **multi-way decision** control statement which is used to choose from several possible alternatives
- The keyword ‘elif’ is short for ‘else if’
- The else statement must always come last, and will be executed as the default action.
- In the case of multiple Boolean expressions, only the first logical Boolean expression which is True will be executed and rest will be skipped.



Keyword

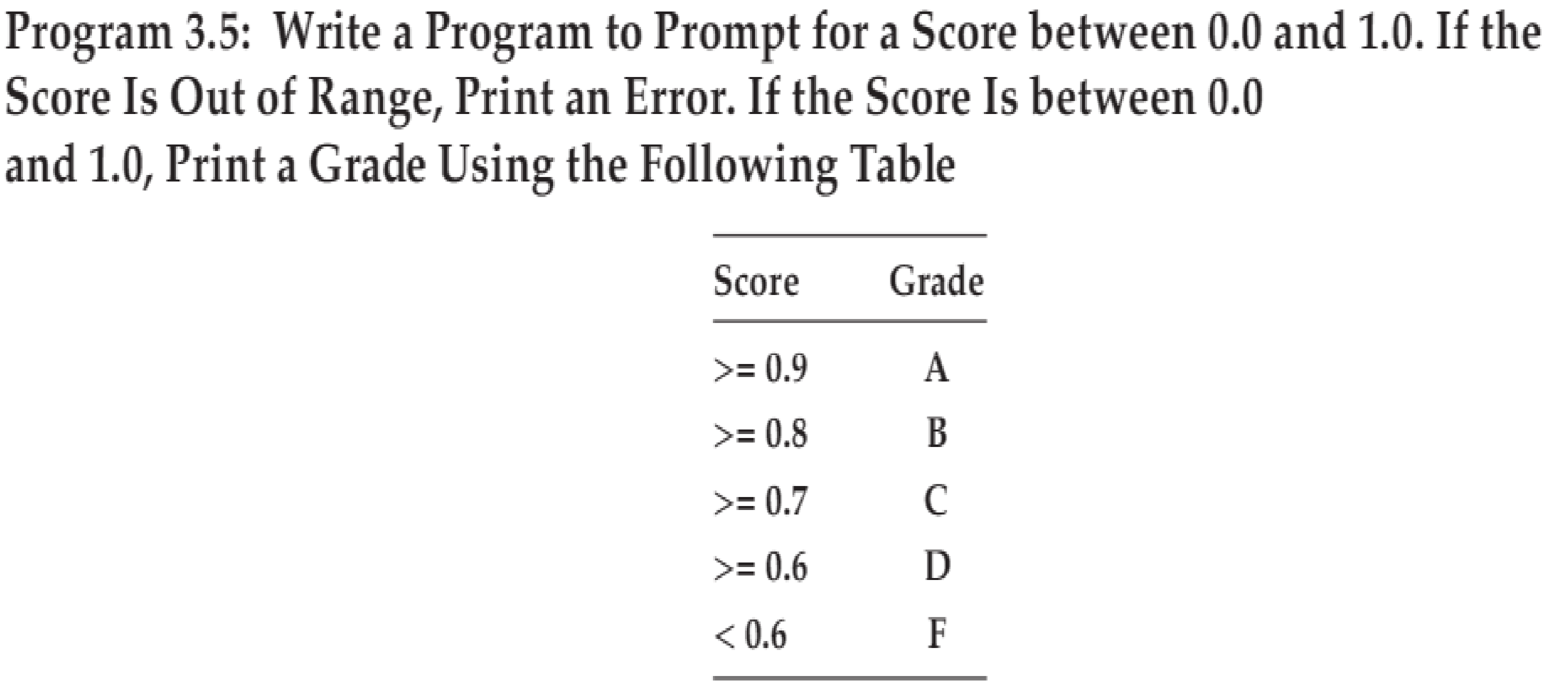
Colon should be present at the end

Indentation

Keyword

```
if Boolean_Expression_1:
     statement_1
elif Boolean_Expression_2:
     statement_2
elif Boolean_Expression_3:
     statement_3
     :
     :
     :
else:
     statement_last
```

- If Boolean_Expression_1 is True, then statement_1 is executed.
- If Boolean_Expression_1 is False and Boolean_Expression_2 is True, then statement_2 is executed.
- If Boolean_Expression_1 and Boolean_Expression_2 are False and Boolean_Expression_3 is True, then statement_3 is executed and so on.
- If none of the Boolean_Expression is True, then statement_last (else block) is executed.

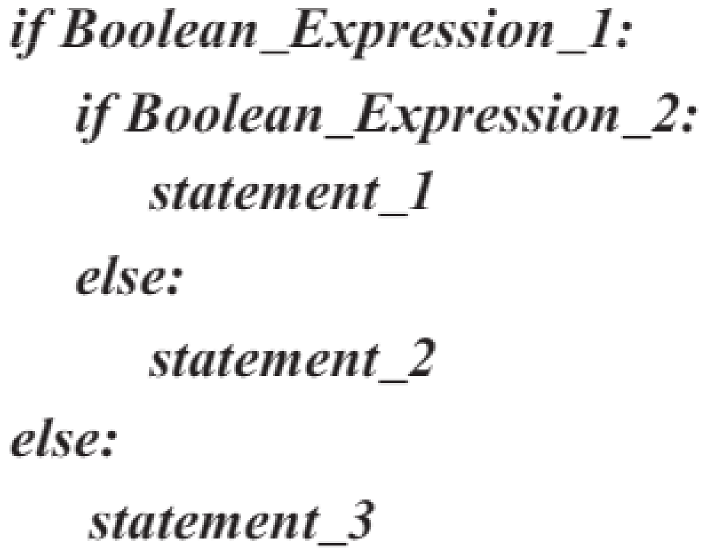


Program 3.5: Write a Program to Prompt for a Score between 0.0 and 1.0. If the Score Is Out of Range, Print an Error. If the Score Is between 0.0 and 1.0, Print a Grade Using the Following Table

Score	Grade
>= 0.9	A
>= 0.8	B
>= 0.7	C
>= 0.6	D
< 0.6	F

Nested If statement (nested conditionals)

- In some situations, you have to place an if statement inside another statement.
- An if statement that contains another if statement either in its if block or else block is called a Nested if statement



```
if Boolean_Expression_1:
    if Boolean_Expression_2:
        statement_1
    else:
        statement_2
else:
    statement_3
```

Example for nested if

```
year = int(input('Enter a year'))
    if year % 4 == 0:
        if year % 100 == 0:
            if year % 400 == 0:
                print('{year} is a Leap Year')
            else:
                print('{year} is not a Leap Year')
        else:
            print('{year} is a Leap Year')
    else:
        print('{year} is not a Leap Year')
```

```
if x == y:
    print('x and y are equal')
else:
    if x < y:
        print('x is less than y')
    else:
        print('x is greater than y')
```

While loop

Syntax :

```
        while  boolean_expression :
                    statement(s)
```

- The while loop starts with the while keyword and ends with a colon. First the Boolean expression is evaluated. If the Boolean expression evaluates to False, then the statements in the while loop block are never executed.
- If the Boolean expression evaluates to True, then the while loop block is executed. After each iteration of the loop block, the Boolean expression is again checked, and if it is True, the loop is iterated again.
- Each repetition of the loop block is called an iteration of the loop.
- This process continues until the Boolean expression evaluates to False and at this point the while statement exits.
- Execution then continues with the first statement after the while loop.

```
#script to print first n natural numbers using while
n = int(input("enter a number"))
if n<=0:
    print(" it is not a natural number")
else:
    i=1
    while i<=n :
        print("current value of i is ",i)
        i=i+1
```

For loop

Syntax:

```
for  iteration_variable   in  sequence :
        statement(s)
```

To generate the sequence range() function is used ,

```
range([start ,] stop [, step])
```

start → value indicates the beginning of the sequence. If the start argument is not specified, then the sequence of numbers start from zero by default.

stop → Generates numbers up to this value but not including the number itself.

step → indicates the difference between every two consecutive numbers in the sequence. The step value can be both negative and positive but not zero

Example:

```
print("All three arguments ''start'', ''stop'' and ''step'' specified in range function")
for i in range(1, 6, 3):
    print(f"{i}")
```

Break and continue

- Both continue and break statements can be used in while and for loops.
- Whenever the break statement is encountered, the execution control immediately jumps to the first instruction following the loop.
- The continue statement is used to pass the control to the next iteration without exiting the loop.

Example on break

```
n = 0
while True:
    print(f"The latest value of n is {n}")
    n = n + 1
    if n > 5:
        print(f"The value of n is greater than 5")
        break
```

Example on continue

```
n = 10
while n > 0:
    print(f"The current value of number is {n}")
    if n == 5:
        print(f"Breaking at {n}")
        n = 10
        continue
    n = n – 1
```

Exit() function

- The exit() is defined in site.py and it works only if the site module is imported so it should be used in the interpreter only.
- It is like a synonym for quit() to make Python more user-friendly.
- It too gives a message when printed

```
# Python program to demonstrate exit()
for i in range(10):
    # If the value of i becomes 5 then the program is forced to exit
    if i == 5:
        # prints the exit message
        print(exit)
        exit()
    print(i)
```