Python Installation

MG ANALYTICS

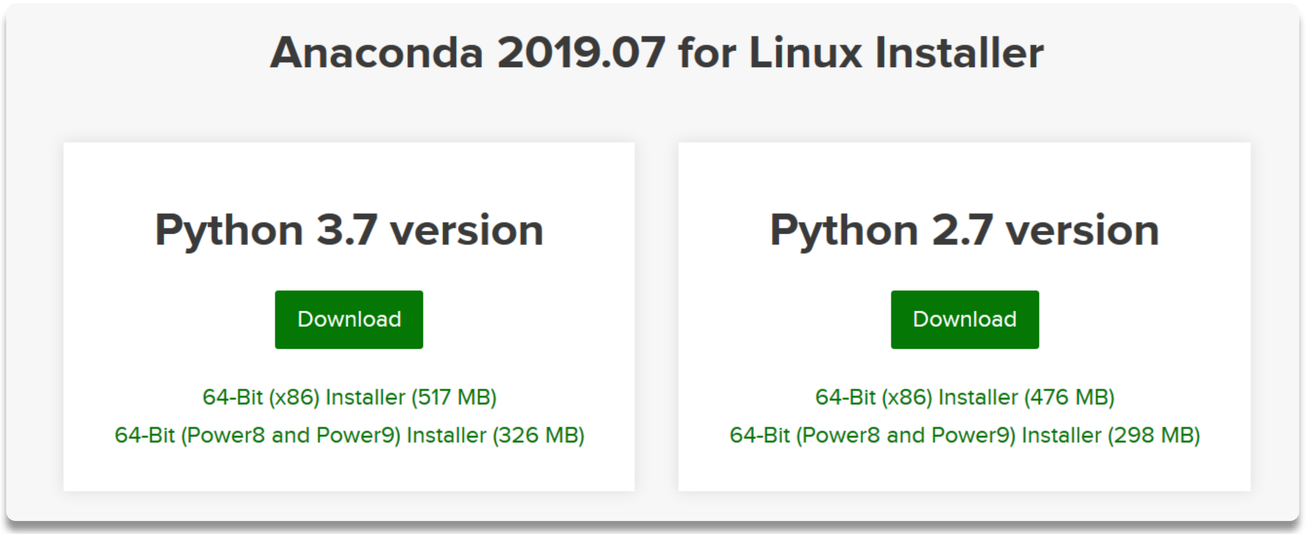
Python 2 vs Python 3

- Python 3 supports modern techniques like AI, machine learning, and data science
- Python 3 is supported by a large Python developer's community. Getting support is easy.
- Its easier to learn Python language compared to earlier versions.
- Offers Powerful toolkit and libraries
- Mixable with other languages

MG ANALYTICS

Environment Setup:

- https://www.anaconda.com/distribution/


Anaconda 2019.07 for Linux Installer

Python 3.7 version

64-Bit (x86) Installer (517 MB)

64-Bit (Power8 and Power9) Installer (326 MB)

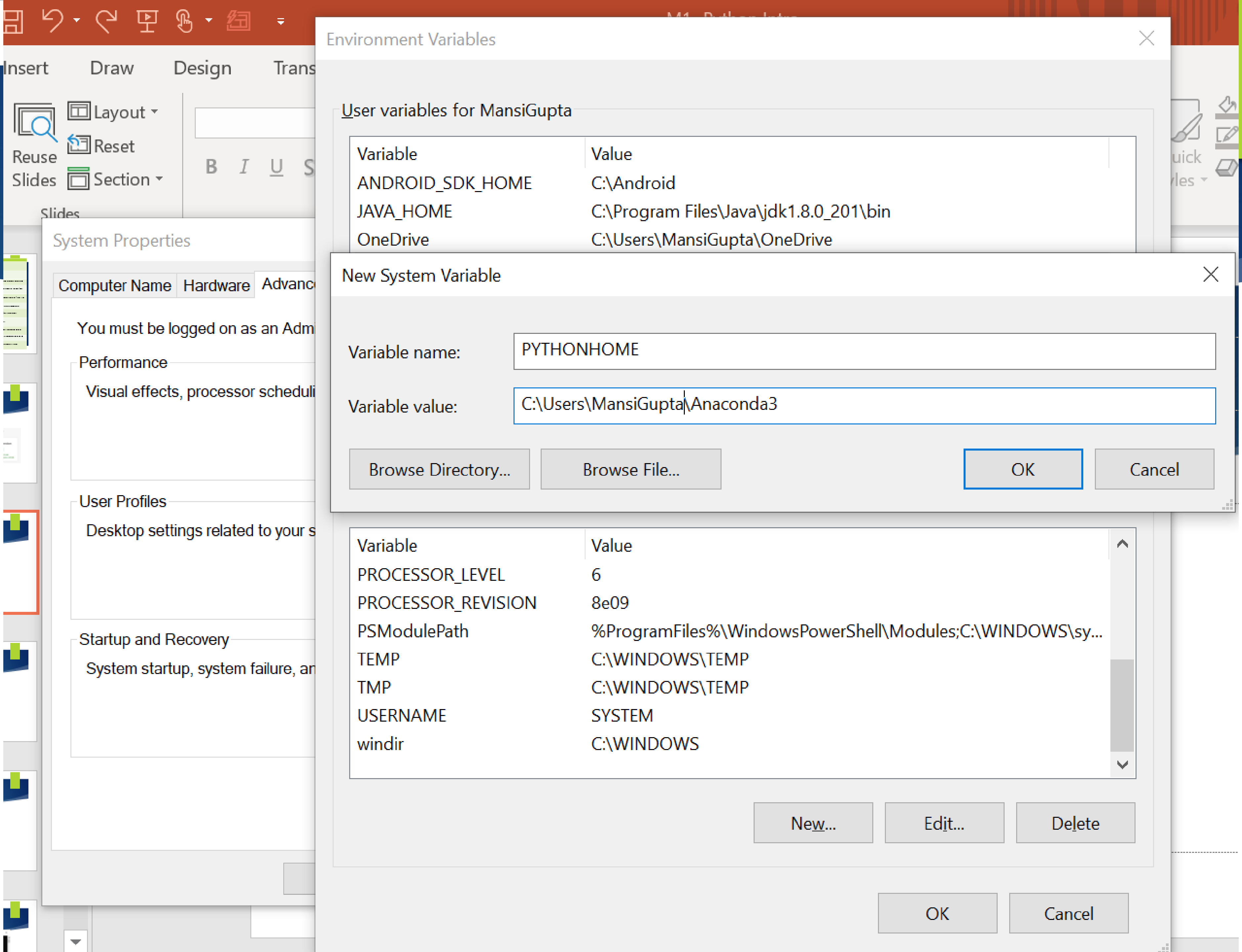
Python 2.7 version

64-Bit (x86) Installer (476 MB)

64-Bit (Power8 and Power9) Installer (298 MB)

MG ANALYTICS

MG ANALYTICS



You must be logged on as an Adm

Visual effects, processor scheduli

Desktop settings related to your s

System startup, system failure, ar

User variables for MansiGupta

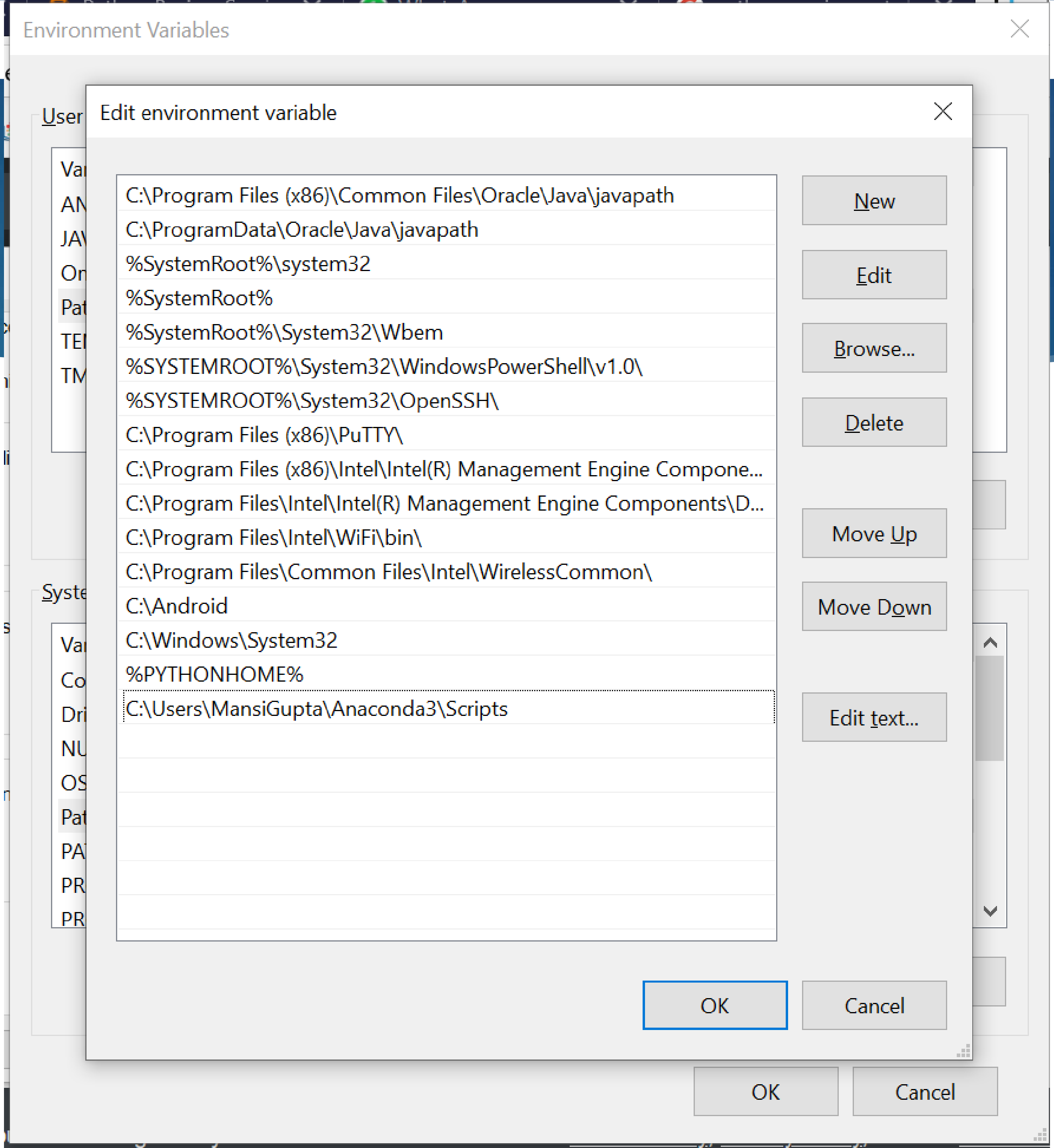
Variable	Value
ANDROID_SDK_HOME	C:\Android
JAVA_HOME	C:\Program Files\Java\jdk1.8.0_201\bin
OneDrive	C:\Users\MansiGupta\OneDrive

Variable name: PYTHONHOME

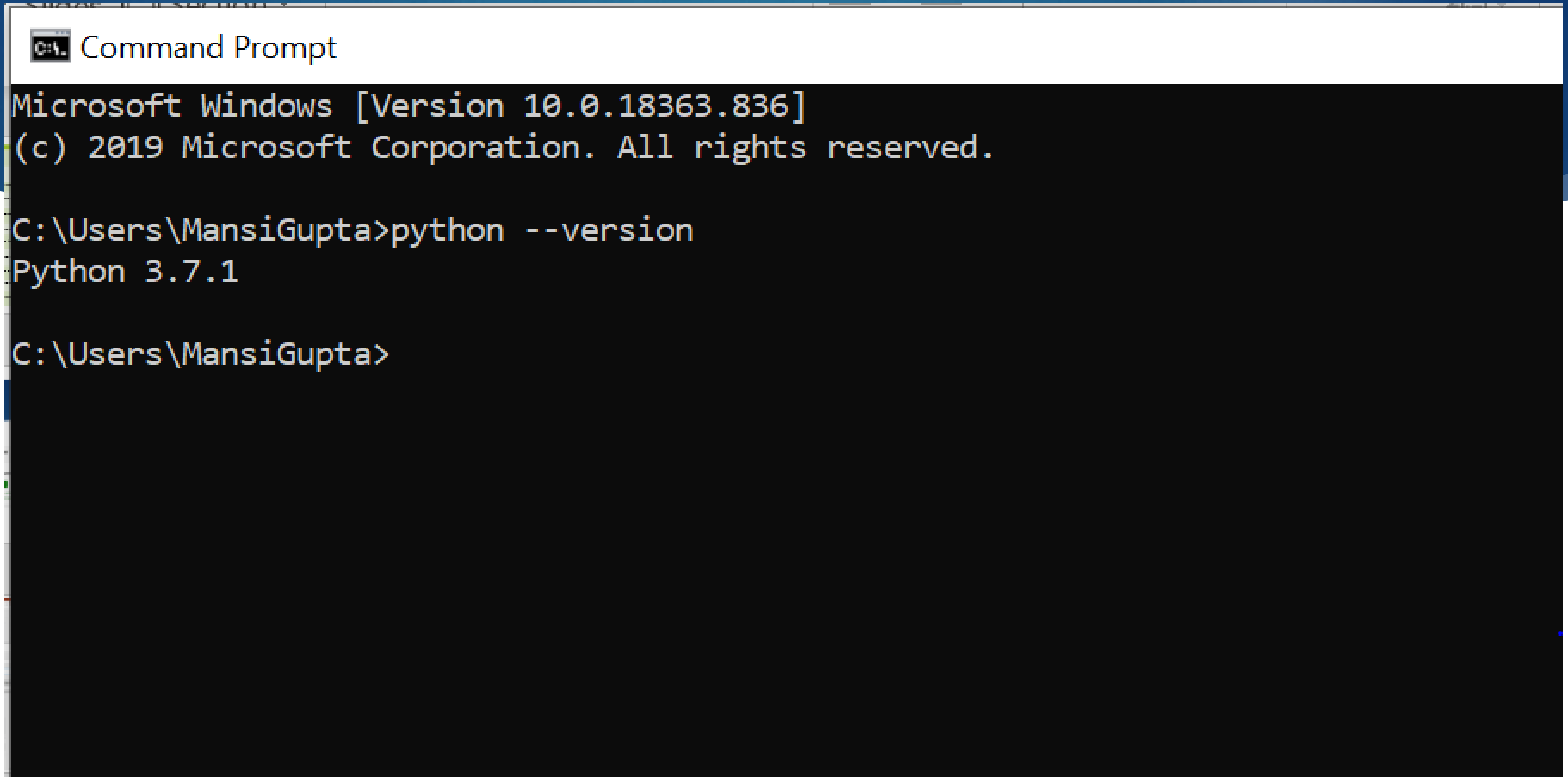
Variable value: C:\Users\MansiGupta\Anaconda3

Variable	Value
PROCESSOR_LEVEL	6
PROCESSOR_REVISION	8e09
PSModulePath	%ProgramFiles%\WindowsPowerShell\Modules;C:\WINDOWS\sy...
TEMP	C:\WINDOWS\TEMP
TMP	C:\WINDOWS\TEMP
USERNAME	SYSTEM
windir	C:\WINDOWS

MG ANALYTICS



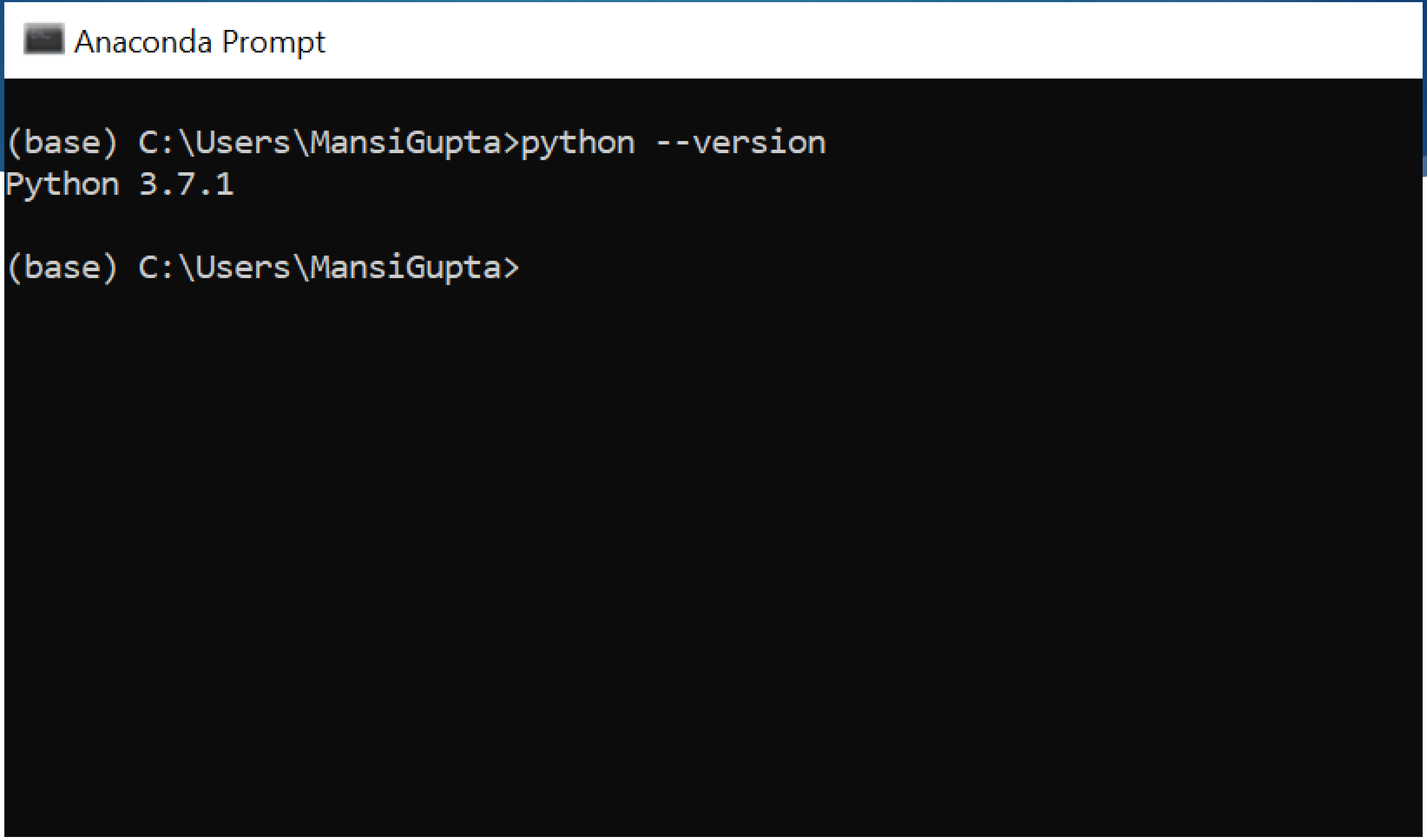
```
C:\Program Files (x86)\Common Files\Oracle\Java\javapath
C:\ProgramData\Oracle\Java\javapath
%SystemRoot%\system32
%SystemRoot%
%SystemRoot%\System32\Wbem
%SYSTEMROOT%\System32\WindowsPowerShell\v1.0\
%SYSTEMROOT%\System32\OpenSSH\
C:\Program Files (x86)\PuTTY\
C:\Program Files (x86)\Intel\Intel(R) Management Engine Compone...
C:\Program Files\Intel\Intel(R) Management Engine Components\D...
C:\Program Files\Intel\WiFi\bin\
C:\Program Files\Common Files\Intel\WirelessCommon\
C:\Android
C:\Windows\System32
%PYTHONHOME%
C:\Users\MansiGupta\Anaconda3\Scripts
```



```
Command Prompt
Microsoft Windows [Version 10.0.18363.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\MansiGupta>python --version
Python 3.7.1

C:\Users\MansiGupta>
```



```
(base) C:\Users\MansiGupta>python --version
Python 3.7.1

(base) C:\Users\MansiGupta>
```

Anaconda Navigator and console

- https://docs.anaconda.com/anaconda/user-guide/tasks/install-packages/
- Spyder
- Jupyter Notebooks
- Pycharm

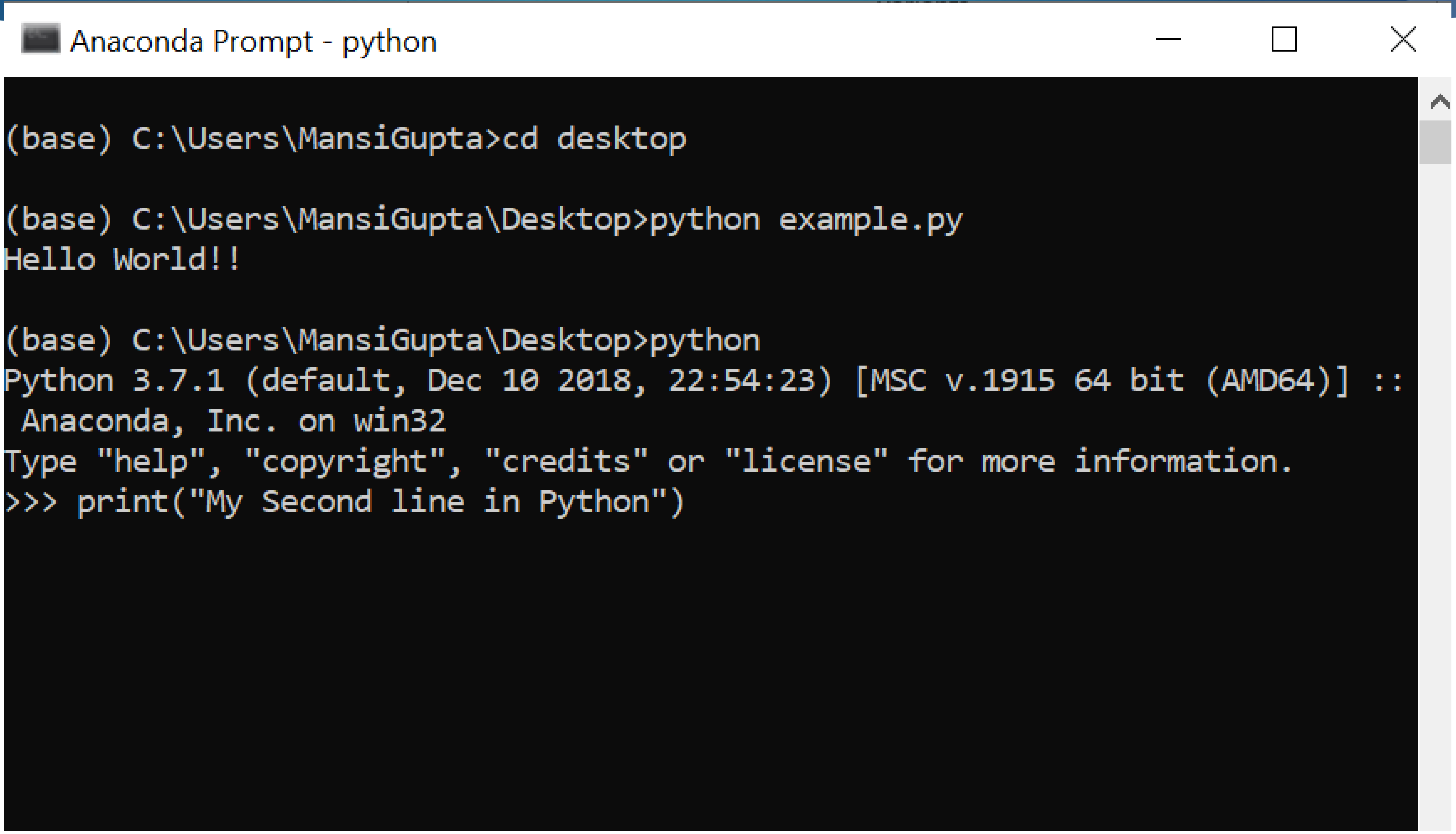
MG ANALYTICS

Jupyter Notebook vs Jupyter Labs

- https://towardsdatascience.com/jypyter-notebook-shortcuts-bf0101a98330

MG ANALYTICS

Running Script using Anaconda Prompt



```
(base) C:\Users\MansiGupta>cd desktop

(base) C:\Users\MansiGupta\Desktop>python example.py
Hello World!!

(base) C:\Users\MansiGupta\Desktop>python
Python 3.7.1 (default, Dec 10 2018, 22:54:23) [MSC v.1915 64 bit (AMD64)] ::
 Anaconda, Inc. on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("My Second line in Python")
```

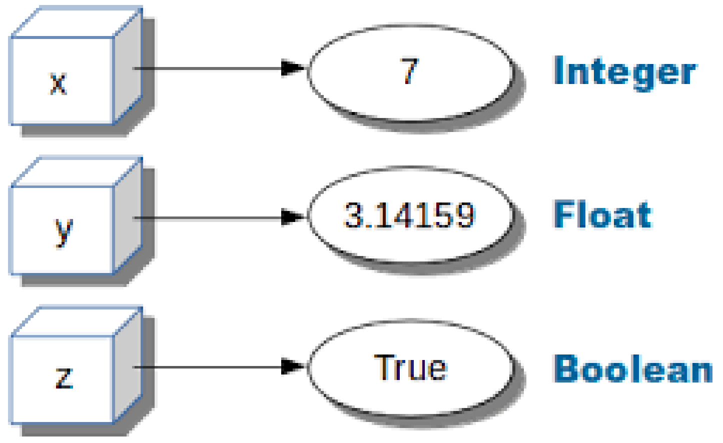
Agenda

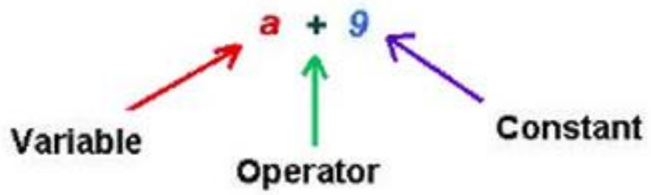
- Running Jupyter Notebook
- Input and Output in Python
- Comments in python
- Data types
- Variables
- Numeric Datatype

Python Data Types

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**

Variables


x
7
Integer
y
3.14159
Float
z
True
Boolean


a + 9
Variable
Operator
Constant

Input Output

PRINT()

INPUT()

Basic Datatypes

- NUMBERS
  - Int
  - float
- STRINGS
  - Strip
  - Case change
  - Formatting
  - Slicing

Agenda - String Operations

- Strip
- Split
- replace
- Case change
- Formatting
- Slicing

List

Comma-separated values (items) between square brackets.
The items in a list need not be of the same type.

- Adding Objects
  - <list> + <list>
  - <list>.append(<object>)
  - <list>.extend(<list>)
- Inserting at a position
  - <list>.insert(<position>, <object>)
- Removing Objects
  - <list>.remove(<object>)
  - <list>.pop()
- Sorting a List
- <list>.sort()
- sorted(<list>, reverse=True)

Boolean

True

False

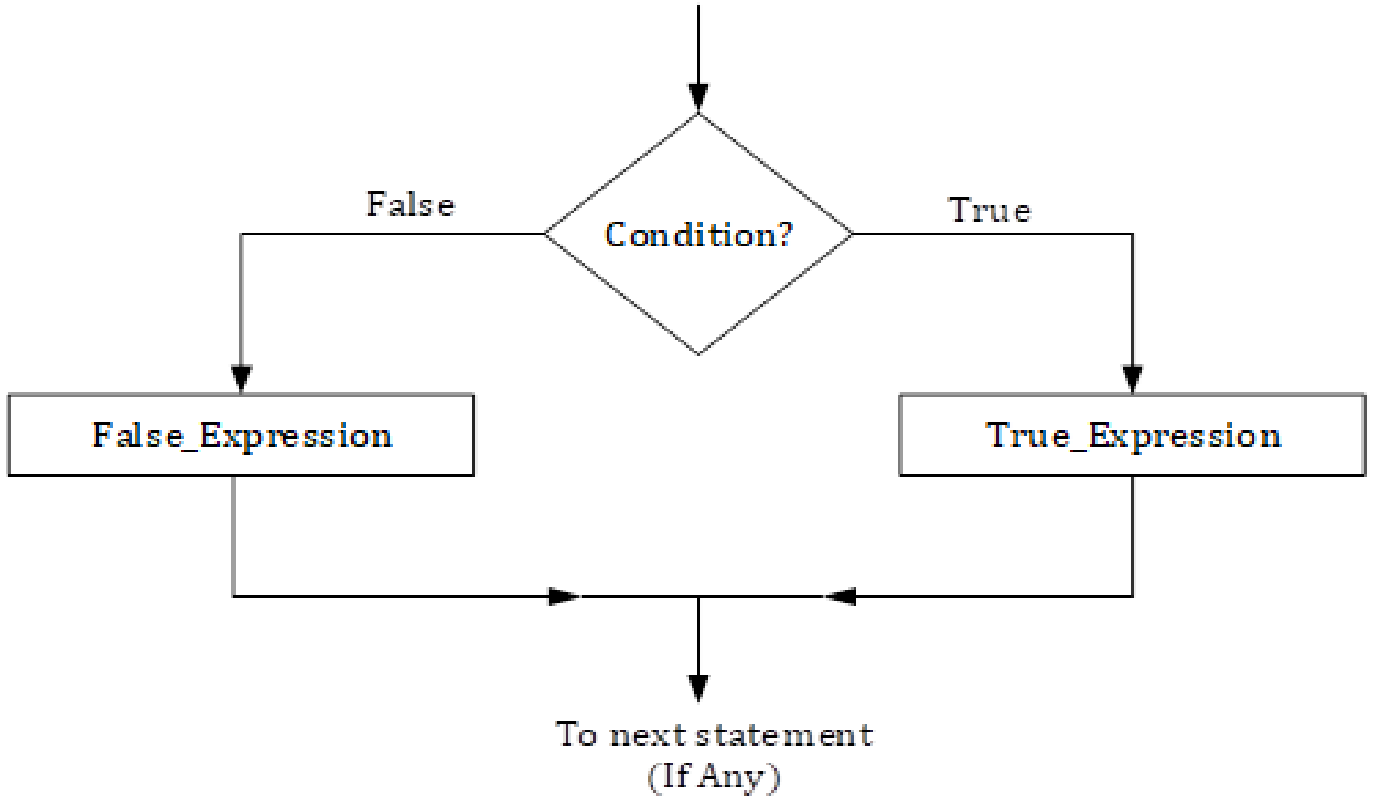
Any condition when evaluated results into Boolean values.

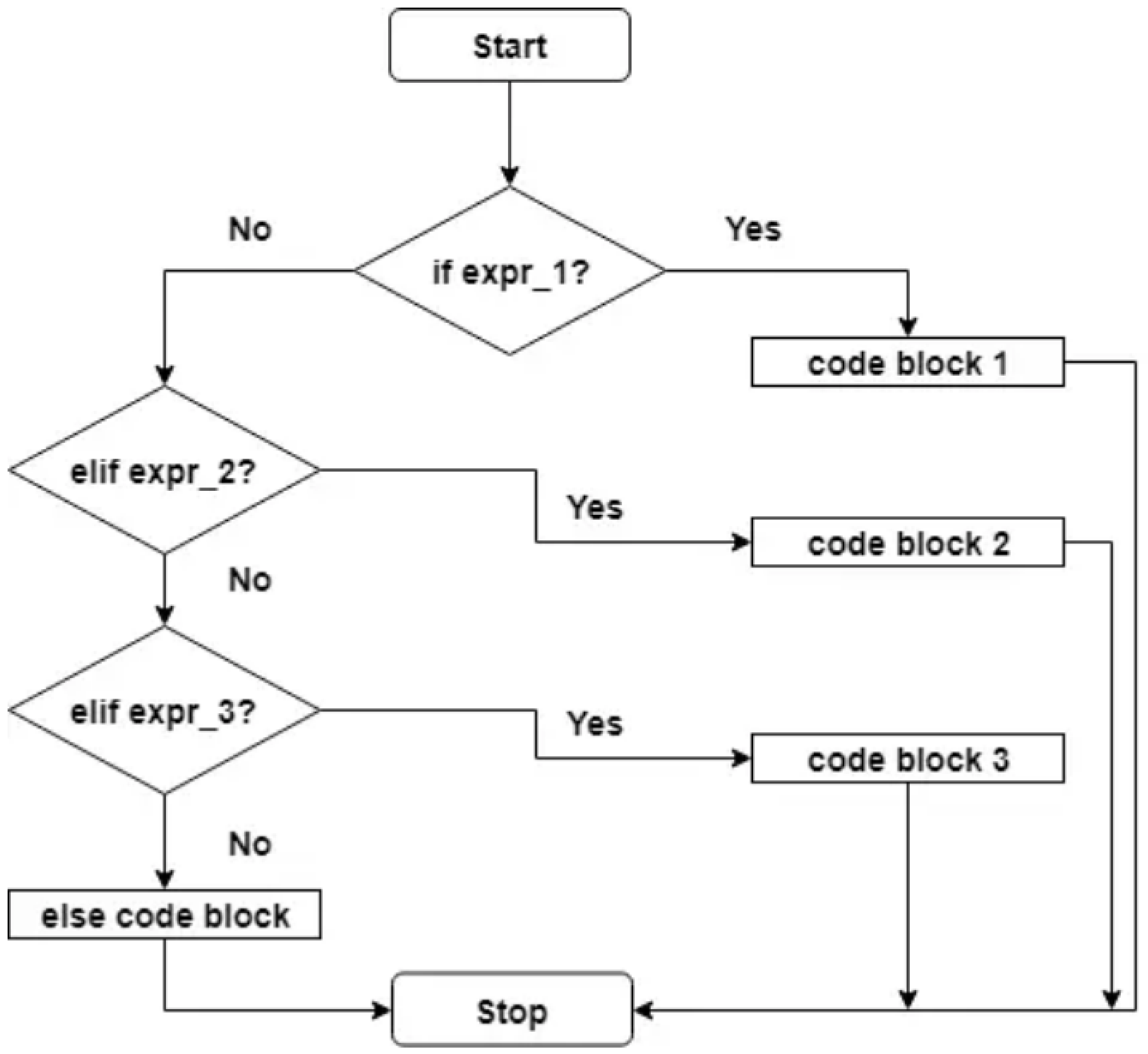
- Boolean type conversion
- Comparison operators
- Chained comparison operators

Decision Making

Any not null or non zero value corresponds to True while zero or null value is

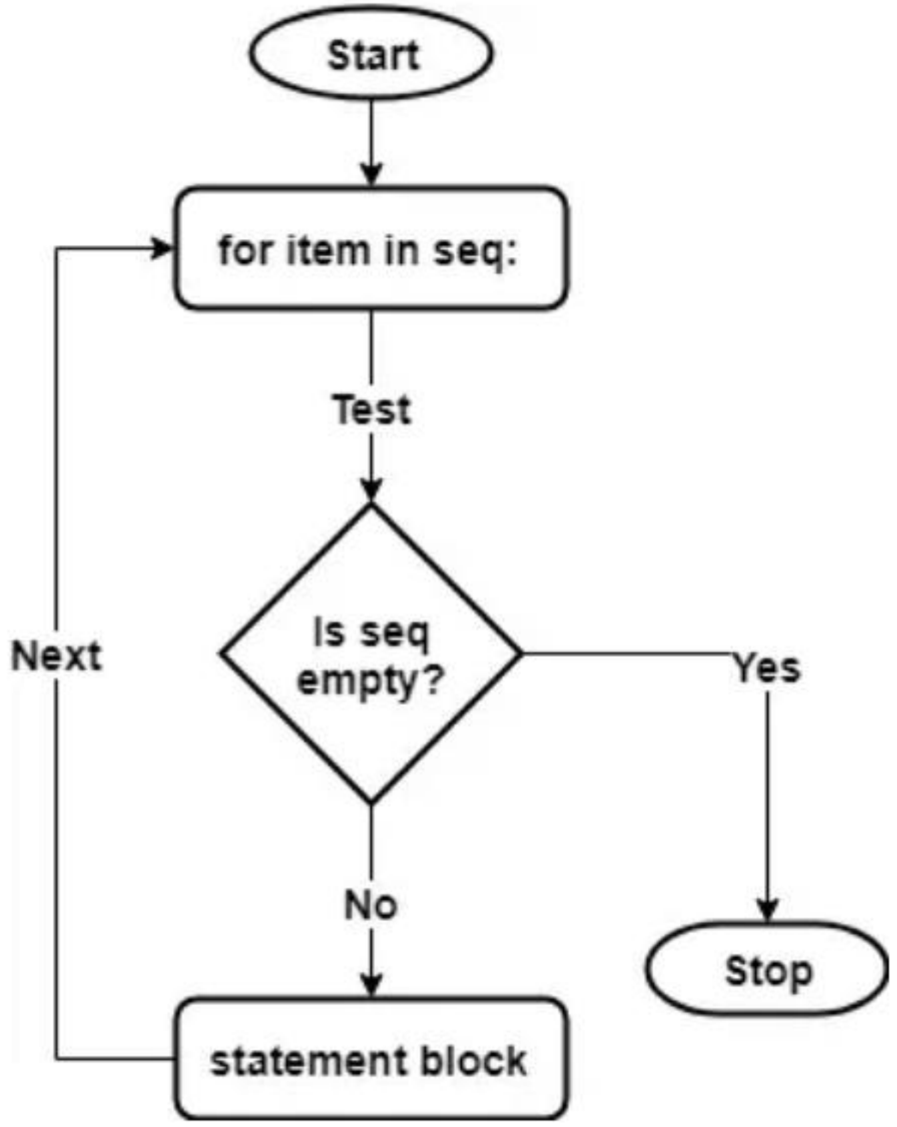
- If condition
- If else condition
- If-elif-else condition




Start
if expr_1?
No
Yes
code block 1
elif expr_2?
Yes
code block 2
No
elif expr_3?
Yes
code block 3
No
else code block
Stop

FOR Loop

Iterates over a sequence of items and runs a statement for each item, once the items are exhausted it stops the loop and runs the next line of code.

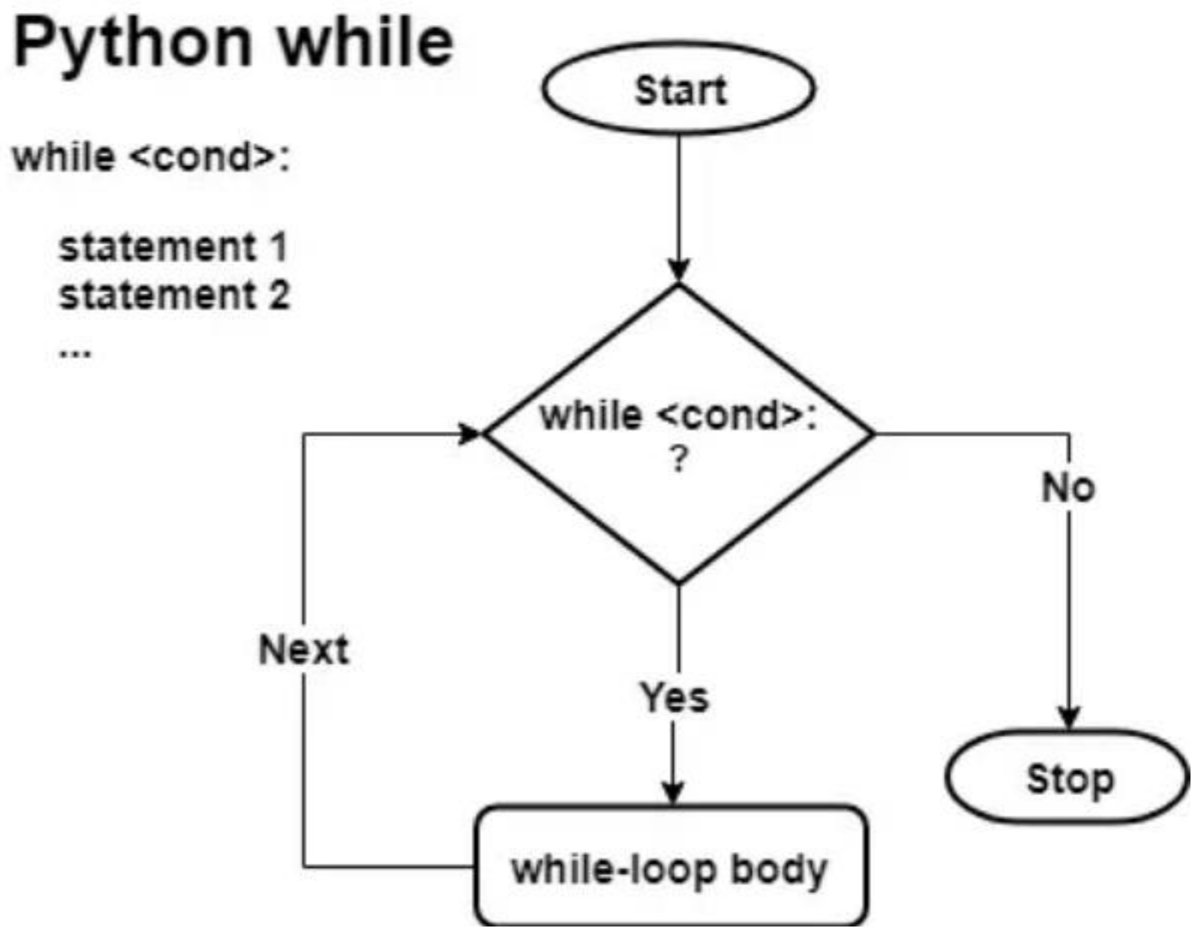


While Loop

It repeatedly executes a block of code indefinite no. of times until the given condition becomes false.

In Python, we can add an optional else clause after the end of "**while**" loop.

The code inside the else clause would always run but after the while loop finishes execution. The one situation when it won't run is if the loop exits after a "**break**" statement.



Python while

```
while <cond>:

    statement 1
    statement 2
    ...
```

Functions

A function is an independent and reusable block of code which you can call any no. of times from any place in a program. It is an essential tool for programmers to split a big project into smaller modules.

List Comprehension

Data Structures

Data organization, management, and storage format that enables efficient access and modification. More precisely, a **data structure** is a collection of **data** values, the relationships among them, and the functions or operations that can be applied to the **data**.

- **Lists** - holds an ordered collection of items
  - i.e. you can store a sequence of items in a list
- **Dictionary** - Like an address-book where you can find the contact details of a person by knowing only his/her name i.e. we associate keys (name) with values (details). Note that the keys must be unique.
- **Tuples** - Similar to lists, but immutable
  - i.e. you cannot modify tuples
- **Sets** - Unordered collections of distinct objects.

Dictionaries

- ✓ It is an unordered collection of Key:value pair, with unique Key values.
- ✓ Keys are unique within a dictionary while values may not be.
- ✓ The values of a dictionary can be of any type
- ✓ the keys must be of an immutable data type such as strings, numbers, or tuples.

- Initialising
  - {“key1” : “value1”, “key2” : “value2”}
  - dict([(“key1”, “value1”), (“key2”, “value2”)])
- Accessing Elements
  - <dict>.keys()
  - <dict>.values()
  - <dict>.items()
  - <dict>[“key”1]

Sets

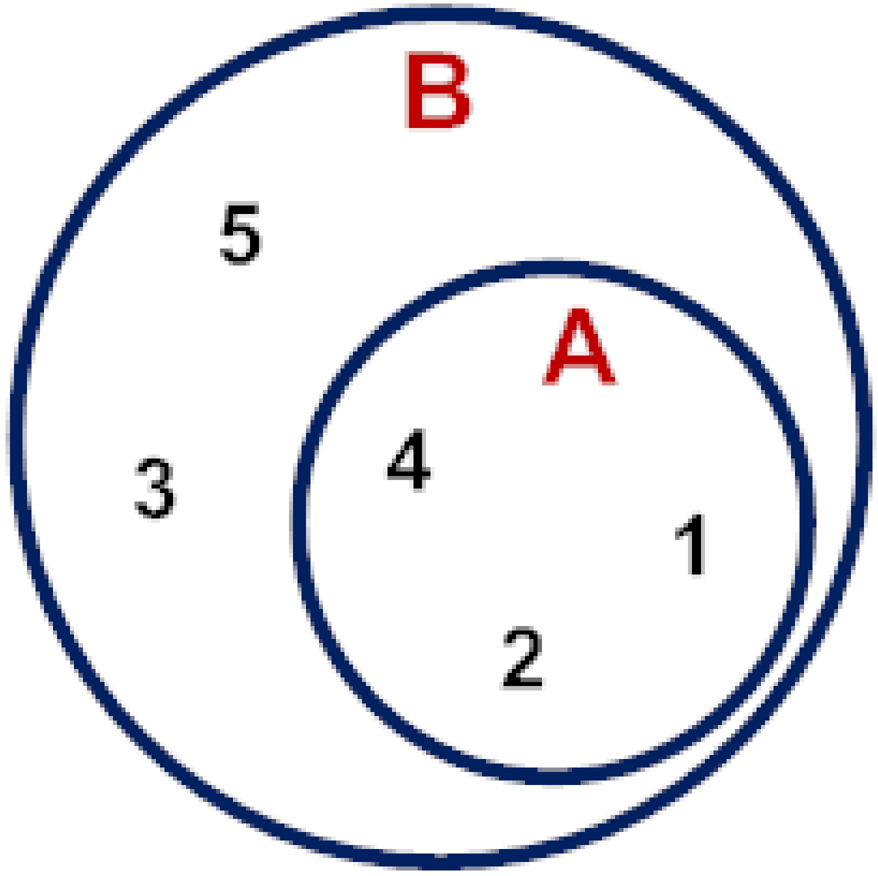
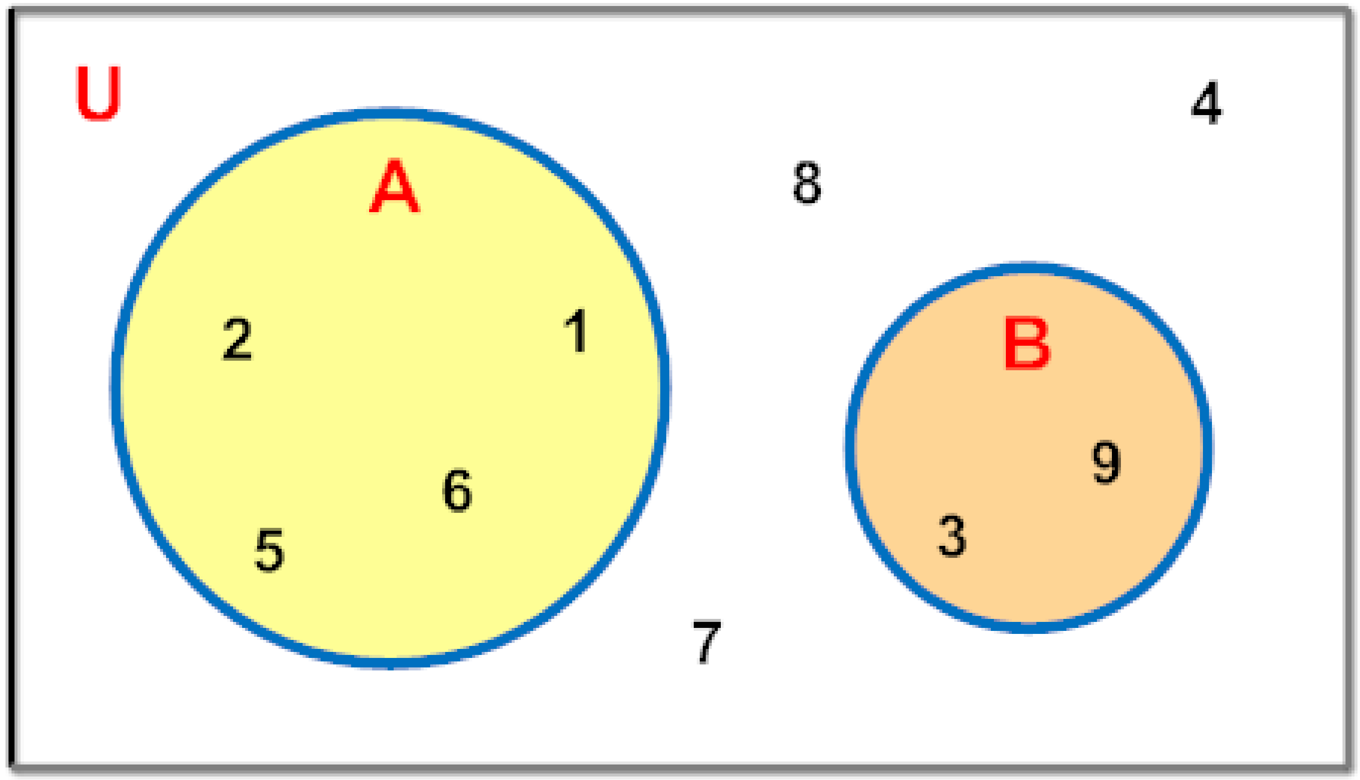
Sets are another type of data structures which can contain multiple elements.

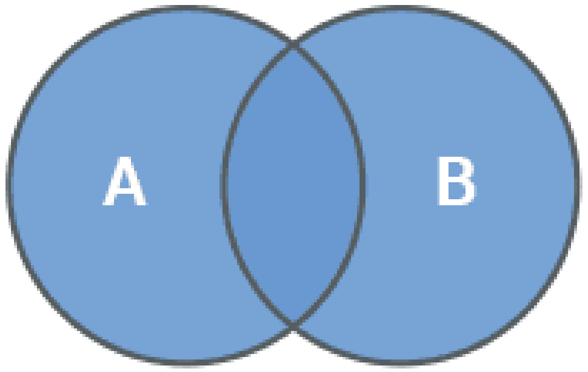
These are unordered like dictionaries

They do not have fixed position like lists.

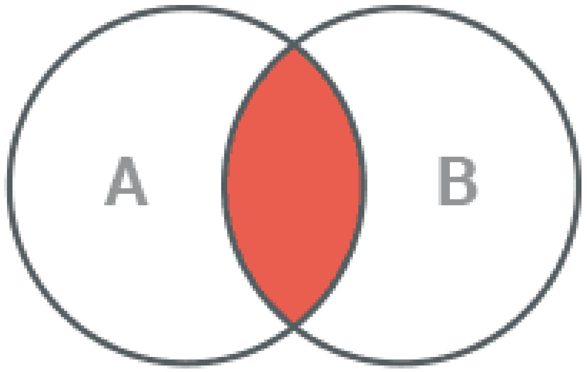
Also elements have to be unique in it.

- defined by “set(<list>)”
- <set>.add(<object>)
- <set>.remove(<object>)
- set.union(<set>, <set>)
- <set>.union(<set>)
- set.intersection(<set>, <set>)
- <set>.intersection(<set>)
- <set1>.difference(<set2>)

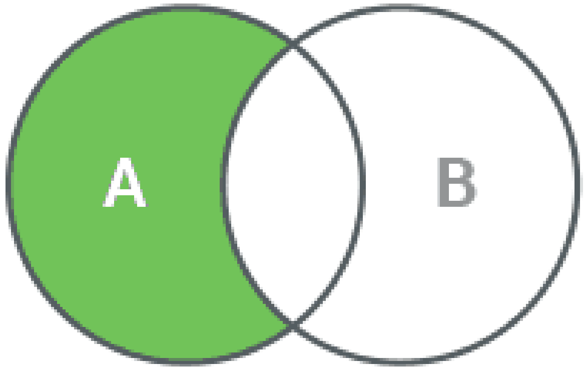

B
5
A
3
4
1
2

U
4
A
8
2
1
B
9
6
3
5
7



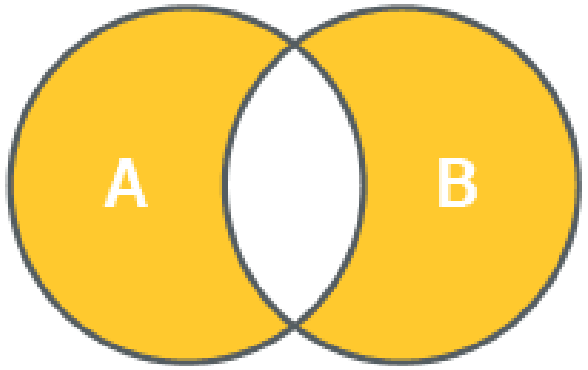
Union



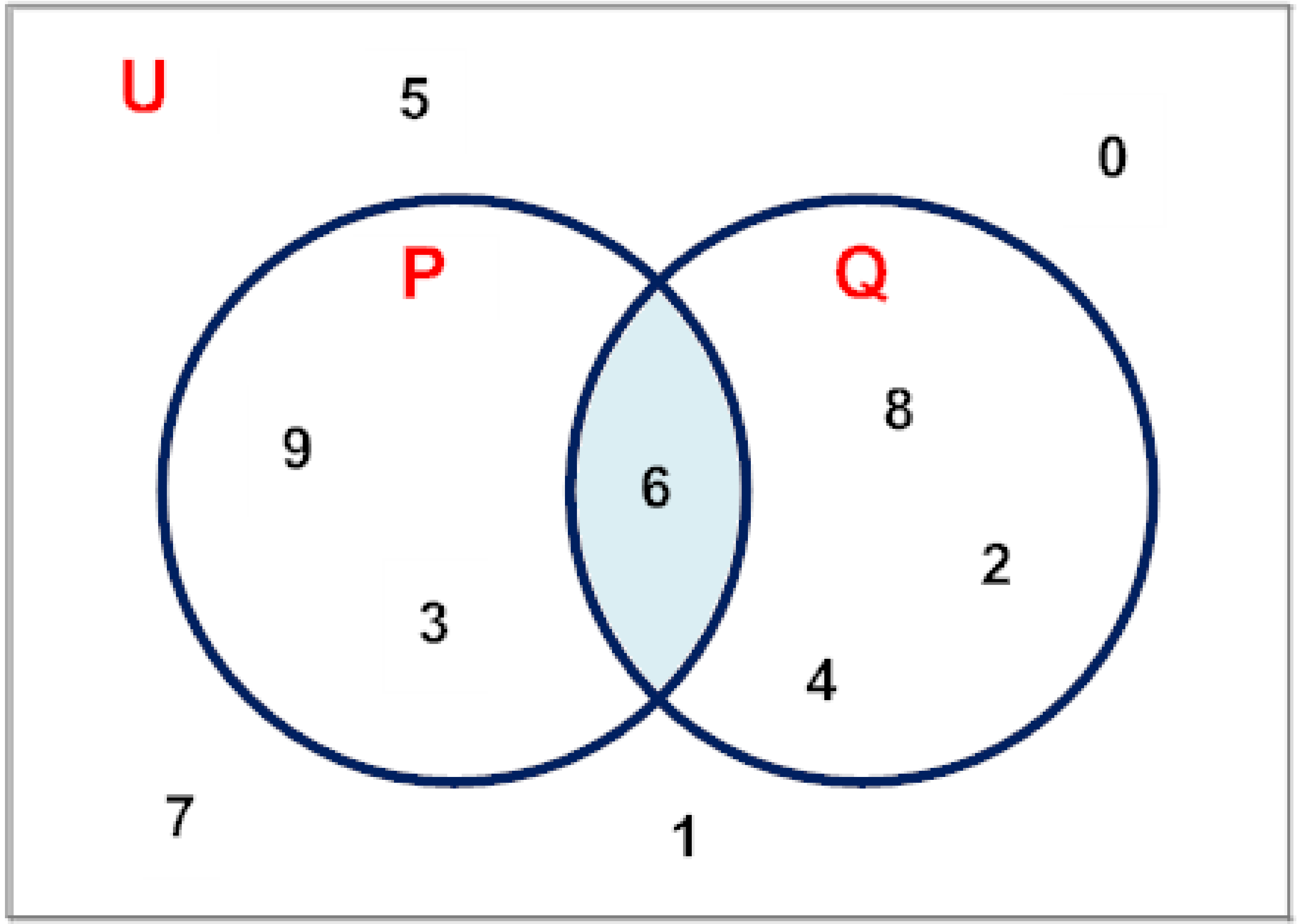
Intersection



Difference



Symmetric Difference


U
5
0
P
Q
9
6
8
2
3
4
7
1

Tuples

Immutable

Tuples can have mutable objects such as lists. So that tuple's elements can not be changed , but it's mutable element's element can be changed

- Defined using
  - tuple(<list>)
  - (<object>, <object>)
- <tuple>.count(<object>)
- <tuple>.index(<object>)