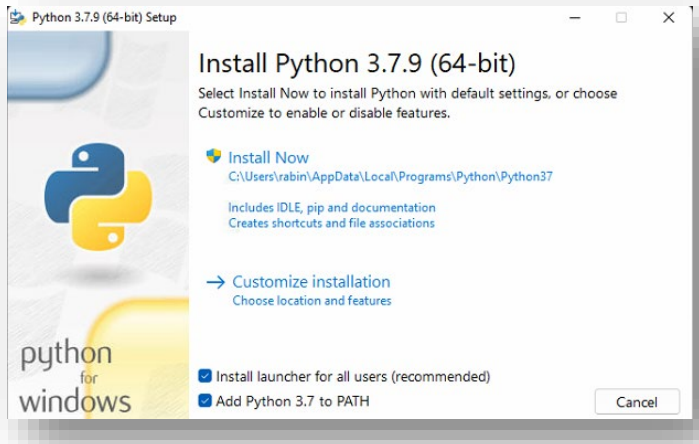
Python 3.7.9 Installation

64 Bit - https://www.python.org/ftp/python/3.7.9/python-3.7.9-amd64.exe

32 Bit - https://www.python.org/ftp/python/3.7.9/python-3.7.9.exe

Check **Add Python 3.7 to PATH**



Understating python virtual environments

Link: https://docs.python.org/3/tutorial/venv.html

1. *Create a virtual environment*

```
C:\>python -m venv py_venv/bn-in
```

2. *Activate the environment*

```
bn-in\Scripts\activate.bat
```

Installing python packages using PIP

Link: https://jupyter.org/install

1. Upgrade pip (for all the environments)

```
pip install --upgrade pip
```

2. Install **Jupyter Notebook** package using **PIP** (In the REAL environment)

```
pip install notebook
```

3. Set the virtual environment kernel in jupyter notebook (In the VIRTUAL environment)

```
pip install --user ipykernel

ipython kernel install --user --name=bn-in
```

Python coding environments

- Using a code editor and file manager
  - Atom: https://atom.io/
  - Notepad++: https://notepad-plus-plus.org/downloads/
- Using Anaconda environment: https://www.anaconda.com/
- Google Collab: https://colab.research.google.com/
- Other IDEs

Python Basics

by Subhrasankar Chatterjee

Why Python?

- Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
- Python has a simple syntax similar to the English language.
- Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
- Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
- Python can be treated in a procedural way, an object-oriented way or a functional way.

Data Types

- Text Type: str
- Numeric Types: int, float, complex
- Sequence Types: list, tuple, range
- Mapping Type: dict
- Set Types: set, frozenset
- Boolean Type: bool
- Binary Types: bytes, bytearray, memoryview
- None Type: NoneType

In [1]:
```
x = 5
print(type(x))
```

```
<class 'int'>
```

Strings

In [2]:
```
a = "Hello"
print(a)
a = "Hello, World!"
print(a[1])
for x in "banana":
    print(x)
a = "Hello, World!"
print(len(a))

txt = "The best things in life are free!"
print("free" in txt)

b = "Hello, World!"
print(b[:5])

a = "Hello, World!"
print(a.upper())

a = " Hello, World! "
print(a.strip()) # returns "Hello, World!"
a = "Hello, World!"
print(a.replace("H", "J"))
a = "Hello, World!"
print(a.split(",")) # returns ['Hello', ' World!']
```

```
Hello
e
b
a
n
a
n
a
13
True
Hello
HELLO, WORLD!
Hello, World!
Jello, World!
['Hello', ' World!']
```

String Methods

Python has a set of built-in methods that you can use on strings.

Note: All string methods return new values. They do not change the original string.

- capitalize() Converts the first character to upper case
- casefold() Converts string into lower case
- center() Returns a centered string
- count() Returns the number of times a specified value occurs in a string
- encode() Returns an encoded version of the string
- endswith() Returns true if the string ends with the specified value
- expandtabs() Sets the tab size of the string
- find() Searches the string for a specified value and returns the position of where it was found
- format() Formats specified values in a string
- format_map() Formats specified values in a string
- index() Searches the string for a specified value and returns the position of where it was found
- isalnum() Returns True if all characters in the string are alphanumeric
- isalpha() Returns True if all characters in the string are in the alphabet
- isdecimal() Returns True if all characters in the string are decimals
- isdigit() Returns True if all characters in the string are digits
- isidentifier() Returns True if the string is an identifier
- islower() Returns True if all characters in the string are lower case
- isnumeric() Returns True if all characters in the string are numeric
- isprintable() Returns True if all characters in the string are printable
- isspace() Returns True if all characters in the string are whitespaces
- istitle() Returns True if the string follows the rules of a title
- isupper() Returns True if all characters in the string are upper case
- join() Joins the elements of an iterable to the end of the string
- ljust() Returns a left justified version of the string
- lower() Converts a string into lower case
- lstrip() Returns a left trim version of the string
- maketrans() Returns a translation table to be used in translations
- partition() Returns a tuple where the string is parted into three parts

- replace() Returns a string where a specified value is replaced with a specified value
- rfind() Searches the string for a specified value and returns the last position of where it was found
- rindex() Searches the string for a specified value and returns the last position of where it was found
- rjust() Returns a right justified version of the string
- rpartition() Returns a tuple where the string is parted into three parts
- rsplit() Splits the string at the specified separator, and returns a list
- rstrip() Returns a right trim version of the string
- split() Splits the string at the specified separator, and returns a list
- splitlines() Splits the string at line breaks and returns a list
- startswith() Returns true if the string starts with the specified value
- strip() Returns a trimmed version of the string
- swapcase() Swaps cases, lower case becomes upper case and vice versa
- title() Converts the first character of each word to upper case
- translate() Returns a translated string
- upper() Converts a string into upper case
- zfill() Fills the string with a specified number of 0 values at the beginning

Python Collections (Arrays)

There are four collection data types in the Python programming language:

- List is a collection which is ordered and changeable. Allows duplicate members.
- Tuple is a collection which is ordered and unchangeable. Allows duplicate members.
- Set is a collection which is unordered, unchangeable*, and unindexed. No duplicate members.
- Dictionary is a collection which is ordered** and changeable. No duplicate members.

Lists

List Items List items are ordered, changeable, and allow duplicate values.

List items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered When we say that lists are ordered, it means that the items have a defined order, and that order will not change.

If you add new items to a list, the new items will be placed at the end of the list.

Note: There are some list methods that will change the order, but in general: the order of the items will not change.

Changeable The list is changeable, meaning that we can change, add, and remove items in a list after it has been created.

Allow Duplicates Since lists are indexed, lists can have items with the same value:

In [3]:
```
mylist = ["apple", "banana", "cherry"]
thislist = ["apple", "banana", "cherry"]
print(thislist)
thislist = ["apple", "banana", "cherry", "apple", "cherry"]
print(thislist)


thislist = ["apple", "banana", "cherry"]
print(len(thislist))

list1 = ["abc", 34, True, 40, "male"]
thislist = list(("apple", "banana", "cherry")) # note the double round-brackets
print(thislist)
```

```
['apple', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'apple', 'cherry']
3
['apple', 'banana', 'cherry']
```

In [4]:
```
thislist = ["apple", "banana", "cherry"]
print(thislist[1])

thislist = ["apple", "banana", "cherry"]
thislist[1] = "blackcurrant"
print(thislist)
```

```
banana
['apple', 'blackcurrant', 'cherry']
```

In [5]:
```
thislist = ["apple", "banana", "cherry"]
thislist.append("orange")
print(thislist)
```

```
thislist = ["apple", "banana", "cherry"]
thislist.insert(1, "orange")
print(thislist)

thislist = ["apple", "banana", "cherry"]
tropical = ["mango", "pineapple", "papaya"]
thislist.extend(tropical)
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.remove("banana")
print(thislist)

thislist = ["apple", "banana", "cherry"]
thislist.pop(1)
print(thislist)

thislist = ["apple", "banana", "cherry"]
del thislist[0]
print(thislist)
```

```
['apple', 'banana', 'cherry', 'orange']
['apple', 'orange', 'banana', 'cherry']
['apple', 'banana', 'cherry', 'mango', 'pineapple', 'papaya']
['apple', 'cherry']
['apple', 'cherry']
['banana', 'cherry']
```

In [6]:
```
thislist = [100, 50, 65, 82, 23]
thislist.sort()
print(thislist)
```

```
[23, 50, 65, 82, 100]
```

List Methods

Python has a set of built-in methods that you can use on lists.

- append() Adds an element at the end of the list
- clear() Removes all the elements from the list
- copy() Returns a copy of the list

- count() Returns the number of elements with the specified value
- extend() Add the elements of a list (or any iterable), to the end of the current list
- index() Returns the index of the first element with the specified value
- insert() Adds an element at the specified position
- pop() Removes the element at the specified position
- remove() Removes the item with the specified value
- reverse() Reverses the order of the list
- sort() Sorts the list

Tuples

Tuple Items Tuple items are ordered, unchangeable, and allow duplicate values.

Tuple items are indexed, the first item has index [0], the second item has index [1] etc.

Ordered When we say that tuples are ordered, it means that the items have a defined order, and that order will not change.

Unchangeable Tuples are unchangeable, meaning that we cannot change, add or remove items after the tuple has been created.

Allow Duplicates Since tuples are indexed, they can have items with the same value:

In [7]:
```
thistuple = ("apple", "banana", "cherry")
print(thistuple)

thistuple = ("apple", "banana", "cherry")
print(thistuple[1])
```

```
('apple', 'banana', 'cherry')
banana
```

In [8]:
```
x = ("apple", "banana", "cherry")
y = list(x)
y[1] = "kiwi"
x = tuple(y)

print(x)
```

```
('apple', 'kiwi', 'cherry')
```

In [9]:
```
## Note: You cannot remove items in a tuple.

thistuple = ("apple", "banana", "cherry")
y = list(thistuple)
y.remove("apple")
thistuple = tuple(y)
```

In [10]:
```
fruits = ("apple", "banana", "cherry")

(green, yellow, red) = fruits

print(green)
print(yellow)
print(red)
```

```
apple
banana
cherry
```

Tuple Methods

Python has two built-in methods that you can use on tuples.

- count() Returns the number of times a specified value occurs in a tuple
- index() Searches the tuple for a specified value and returns the position of where it was found

Sets

Set Items Set items are unordered, unchangeable, and do not allow duplicate values.

Unordered Unordered means that the items in a set do not have a defined order.

Set items can appear in a different order every time you use them, and cannot be referred to by index or key.

Unchangeable Set items are unchangeable, meaning that we cannot change the items after the set has been created.

Once a set is created, you cannot change its items, but you can remove items and add new items.

Duplicates Not Allowed Sets cannot have two items with the same value.

In [11]:
```
thisset = {"apple", "banana", "cherry"}
print(thisset)
```

```
{'apple', 'banana', 'cherry'}
```

In [12]:
```
#You cannot access items in a set by referring to an index or a key.


thisset = {"apple", "banana", "cherry"}

for x in thisset:
    print(x)
```

```
apple
banana
cherry
```

In [13]:
```
thisset = {"apple", "banana", "cherry"}
thisset.add("orange")
print(thisset)

thisset = {"apple", "banana", "cherry"}
thisset.remove("banana")
print(thisset)

thisset = {"apple", "banana", "cherry"}
thisset.discard("banana")
print(thisset)
```

```
{'apple', 'orange', 'banana', 'cherry'}
{'apple', 'cherry'}
{'apple', 'cherry'}
```

In [14]:
```
set1 = {"a", "b" , "c"}
set2 = {1, 2, 3}

set3 = set1.union(set2)
print(set3)
```

```
{1, 'c', 2, 3, 'a', 'b'}
```

Set Methods

Python has a set of built-in methods that you can use on sets.

- add() Adds an element to the set
- clear() Removes all the elements from the set
- copy() Returns a copy of the set
- difference() Returns a set containing the difference between two or more sets
- difference_update() Removes the items in this set that are also included in another, specified set
- discard() Remove the specified item
- intersection() Returns a set, that is the intersection of two other sets
- intersection_update() Removes the items in this set that are not present in other, specified set(s)
- isdisjoint() Returns whether two sets have a intersection or not
- issubset() Returns whether another set contains this set or not
- issuperset() Returns whether this set contains another set or not
- pop() Removes an element from the set
- remove() Removes the specified element
- symmetric_difference() Returns a set with the symmetric differences of two sets
- symmetric_difference_update() inserts the symmetric differences from this set and another
- union() Return a set containing the union of sets

Dictionary

Ordered or Unordered? As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.

When we say that dictionaries are ordered, it means that the items have a defined order, and that order will not change.

Unordered means that the items does not have a defined order, you cannot refer to an item by using an index.

Changeable Dictionaries are changeable, meaning that we can change, add or remove items after the dictionary has been created.

Duplicates Not Allowed Dictionaries cannot have two items with the same key:

In [15]:
```
thisdict = {
```

```
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964,
  "year": 2020
}
print(thisdict)
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 2020}
```

In [16]:
```
print(len(thisdict))
```

```
3
```

In [17]:
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
x = thisdict["model"]

x = thisdict.get("model")

x = thisdict.keys()
```

In [18]:
```
car = {
"brand": "Ford",
"model": "Mustang",
"year": 1964
}

x = car.keys()

print(x) #before the change

car["color"] = "white"

print(x) #after the change
```

```
dict_keys(['brand', 'model', 'year'])
dict_keys(['brand', 'model', 'year', 'color'])
```

In [19]:
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
```

```
  "year": 1964
}
thisdict["year"] = 2018

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"year": 2020})
```

In [20]:
```
thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict["color"] = "red"
print(thisdict)

thisdict = {
  "brand": "Ford",
  "model": "Mustang",
  "year": 1964
}
thisdict.update({"color": "red"})
```

```
{'brand': 'Ford', 'model': 'Mustang', 'year': 1964, 'color': 'red'}
```

In [21]:
```
for x in thisdict:
    print(thisdict[x])

for x in thisdict.values():
    print(x)

for x in thisdict:
    print(x)

for x, y in thisdict.items():
    print(x, y)
```

```
Ford
Mustang
1964
red
Ford
Mustang
1964
red
brand
model
year
color
brand Ford
model Mustang
year 1964
color red
```

Dictionary Methods

Python has a set of built-in methods that you can use on dictionaries.

- clear() Removes all the elements from the dictionary
- copy() Returns a copy of the dictionary
- fromkeys() Returns a dictionary with the specified keys and value
- get() Returns the value of the specified key
- items() Returns a list containing a tuple for each key value pair
- keys() Returns a list containing the dictionary's keys
- pop() Removes the element with the specified key
- popitem() Removes the last inserted key-value pair
- setdefault() Returns the value of the specified key. If the key does not exist: insert the key, with the specified value
- update() Updates the dictionary with the specified key-value pairs
- values() Returns a list of all the values in the dictionary

1. Write a program to compute and print the taxi fare based on the following chart. Total number of Kilometers traveled will be input by the user.

First 12 KM:	Rs. 100/-
Next 4 KM:	Rs. 8 / KM
Next 4 KM:	Rs. 6 / KM
Above 20 KM:	Rs. 5 / KM

2. Acceleration due to gravity of a celestial object of mass M and radius R is given by

$$g = G\frac{M}{R^2}$$

Calculate and print the values of g's for the earth and moon, given that

- Mass of the earth = 5.972×10^{24} kg
- Radius of the earth = 6361 km
- Mass of the moon = 7.35×10^{22} kg
- Radius of the moon = 1737 km
- Assume the value of Newton's constant G as 6.67408×10^{-11} $m^3 \cdot kg^{-1} \cdot s^{-2}$

3. Write a program to find the sum of the following series for a given value of n.

$$S = 1 - \frac{1}{3} + \frac{1}{3^2} - \frac{1}{3^3} + \cdots (-1)^n \frac{1}{3^n}$$

Your program should take n as input from the user and print the sum.

4. An Armstrong number is the number which is the sum of the cubes of all its units, tens and hundred digits, etc. For example, for a three-digit number 153,

$$153 = 1*1*1 + 5*5*5 + 3*3*3 = 1 + 125 + 27 = 153$$

For a four-digit number 1634,

$$1634 = 1*1*1*1 + 6*6*6*6 + 3*3*3*3 + 4*4*4*4 = 1 + 1296 + 81 + 256 = 1634$$

Write a Python Program to check if the user entered number is Armstrong number or not.

5. A number is called Harshad number (also called Niven number) if the number is divisible by the sum of its digits. For example, 210 is a Harshad number because 210 is divisible by the sum of its digits $(2 + 1 + 0 = 3)$. Write a program which will print the first 10 Harshad numbers with n-digits. The number n will be known at the time of running your program.

6. Write a Python Program to find whether a string or number is palindrome or not, and print the decision as output.

7. Evaluate the expression of T_p, and print the result.

$$T_p = T_s\sqrt{\frac{R_s\sqrt{\frac{1-\propto}{\sigma}}}{2D}}$$

Where, $\alpha = 0.306, Ts = 6.96 \times 10^8\, m,\ Rs = 6.96 \times 10^8\, m,\ D = 1.496 \times 10^{11}\, m\ and\ \sigma = 1.2$

8. Symmetric matrix is a square matrix which is equal to its transpose. If $A[][]$ is a square matrix with $(n\ x\ n)$ order, then this matrix is said to be symmetric if every element at i^{th} row and j^{th} column is equal to element at j^{th} row and i^{th} column, that is, $A[i][j] == A[j][i]$. Your program should take the input matrix from the user, display it and check whether the matrix is symmetric or not.

9. Saddle point of a matrix is an element in the matrix which is smallest in its row and largest in its column. For example, in the following matrix, 7 is the saddle point at (2,2).

6	3	1
9	7	8
2	4	5

Write a program to find all the saddle point in a given matrix

10. A ball is released from a height of h meters. Each time it bounces on the floor, its velocity becomes halved. Write a program, which takes the value of h and then prints the total distance traversed by the ball when it touches the ground for the n^{th} time. Assume that the value of acceleration due to gravity, g is 9.8 m/sec^2.