Hello, World!

This is my first program using Jupyter Notebook.

In [1]:
```
print("Hello, World!")
```

```
Hello, World!
```

Equation format using Markdown:

$a = b + c$

Variables

In [2]:
```
x = 3
```

Show variables in the workspace

In [3]:
```
%whos
```

```
Variable   Type    Data/Info
----------------------------
x          int     3
```

In [4]:
```
print(type(x))
```

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

In [5]:
```
x = 2.71
```

In [6]:
```
print(type(x))
```

```
<class 'float'>
```

In [7]:
```
y = 26
```

In [8]:
```
%whos
```

```
Variable   Type     Data/Info
-----------------------------
x          float    2.71
y          int      26
```

In [9]:
```
a, b, c, d, e = 4, 7, 8.4, 1, -5
```

In [10]:
```
%whos
```

```
Variable   Type     Data/Info
----------------------------
a          int      4
b          int      7
c          float    8.4
d          int      1
e          int      -5
x          float    2.71
y          int      26
```

In [11]:
```
del y
```

In [12]:
```
%whos
```

```
Variable   Type     Data/Info
----------------------------
a          int      4
b          int      7
c          float    8.4
d          int      1
e          int      -5
x          float    2.71
```

In [13]:
```
c = 2 + 4j
```

In [14]:
```
print(type(c))
```

```
<class 'complex'>
```

In [15]:
```
s = "How are you?"
```

In [16]:
```
print(type(s))
```

```
<class 'str'>
```

In [17]:
```
_g = 6  # variable can start with _
```

Operators

In [18]:
```
10 // 3 # floor division
```

Out[18]:
```
3
```

In [19]:
```
a = 2
```

In [20]:
```
b = 3.0
```

In [21]:
```
a + b
```

Out[21]:
```
5.0
```

In [22]:
```
b - a
```

Out[22]:
```
1.0
```

In [23]:
```
(a**b)/4
```

Out[23]:
```
2.0
```

In [24]:
```
s1 = "hello"
s2 = "world"
s1 + s2
```

Out[24]:
```
'helloworld'
```

In [25]:
```
3 % 2
```

Out[25]:
```
1
```

Bool

In [26]:
```
a = True
b = True
c = False
```

In [27]:
```
print(a and b)
print(a and c)
print(not(a))
print(a or c)
```

```
True
False
False
True
```

Comparisons

In [28]:
```
print(2 < 3)
print(2 == 3)
print(2 != 3)
```

```
True
False
True
```

Functions

round(x, y) round to the nearest integer

In [29]:
```
print(round(5.6231))
```

```
6
```

Round with 3 decimals

In [30]:
```
print(round(5.6231, 3))
```

```
5.623
```

divmod(x, y) outputs the quotient and the remainder in a tuple

In [31]:
```
divmod(27, 5)
```

Out[31]:
```
(5, 2)
```

In [32]:
```
print(type(divmod(34, 9)))
```

```
<class 'tuple'>
```

isinstance() returns True if the first argument is an instance of that class

In [33]:
```
isinstance(1, int)
```

Out[33]:
```
True
```

In [34]:
```
isinstance(1.0, int)
```

Out[34]:
```
False
```

Check if the first argument is an integer or a float.

In [35]:
```
isinstance(1.0, (int, float))
```

Out[35]:
```
True
```

In [36]:
```
isinstance(2 + 3j, (int, float))
```

Out[36]:
```
False
```

In [37]:
```
isinstance(2 + 3j, (int, float, complex))
```

Out[37]:
```
True
```

pow(x, y, z) x raise to the power y and remainder by z: $x^y \% z$

In [38]:
```
pow(2, 4)
```

Out[38]:
```
16
```

In [39]:
```
pow(2, 4, 7)
```

Out[39]:
```
2
```

input() enter value

In [40]:
```
x = input("Enter a number: ")
```

```
Enter a number: 14
```

In [41]:
```
type(x)
```

Out[41]:
```
str
```

In [42]:
```
x = int(input("Enter a number: "))
```

```
Enter a number: 37
```

In [43]:
```
type(x)
```

Out[43]:
```
int
```

In [44]:
```
a = float(input("Enter a float: "))
```

```
Enter a float: 9.41
```

In [45]:
```
type(a)
```

Out[45]:
```
float
```

Get help by typing an ?. For instance:

In [46]:
```
pow?
```

In [47]:
```
help(pow)
```

```
Help on built-in function pow in module builtins:

pow(x, y, z=None, /)
    Equivalent to x**y (with two arguments) or x**y % z (with three arguments)

    Some types, such as ints, are able to use a more efficient algorithm when
    invoked using the three argument form.
```

Conditional

Take two integers and print the bigger number.

In [48]:
```
a = int(input("Input first integer, a = "))
b = int(input("Input second integer, b = "))
if a > b:
    print(f"a = {a} is greater than b = {b}.")
elif a == b:
    print(f"a = {a} is equal to b = {b}.")
else:
    print(f"a = {a} is less than b = {b}.")
```

```
Input first integer, a = 5
Input second integer, b = 7
a = 5 is less than b = 7.
```

In [49]:
```
"""
1- User enter a floating point number.
2- Find the integer portion of the number
3- Check if the integer portions is even or not.
"""

num = float(input("Enter a real number: "))
num = int(num // 1)
if num < 0:
    num += 1
if num % 2 == 0:
    print(f"{num} is even.")
else:
    print(f"{num} is odd.")
```

```
Enter a real number: 8.1
8 is even.
```

Loops

In [50]:
```
n = 5
counter = 1
while counter <= n:
    print(counter)
    counter += 1
```

```
1
2
3
4
5
```

In [51]:
```
i = 1
while True:
    if i % 17 == 0:
        print('break')
        break
    else:
        i += 1
        continue
    print("I'm inside the loop")
print('done')
```

```
break
done
```

In [52]:
```
L = []
for i in range(10):
    print(i)
    L.append(i**2)
print(L)
```

```
0
1
2
3
4
5
6
7
8
9
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [53]:
```
L = []
for i in range(4, 10, 2):
    print(i)
    L.append(i**2)
print(L)
```

```
4
6
8
[16, 36, 64]
```

In [54]:
```
S = {"orange", "apple", "grape", 5.8}
print(S)
for x in S:
    print(x)
else:
    print("Loop complete.")
```

```
{5.8, 'orange', 'grape', 'apple'}
5.8
orange
grape
apple
Loop complete.
```

In [55]:
```
D = {"apple": 44, "cherry": "red"}
for x in D:
    print(x, D[x])
```

```
apple 44
cherry red
```

Functions

In [56]:
```
"""
Givem a list of numbers, make another list sorted in non-decreasing order.
"""
import random

# Input
n = 10
L = []
for x in range(n):
    y = random.randint(-20, 30)
    L.append(y)

# Code - Insertion Sort
for j in range(0, len(L)):
    key = L[j]
    i = j - 1
    while i >= 0 and L[i] > key:
        L[i+1] = L[i]
        i = i - 1
    L[i+1] = key

# Output
print(L)
```

```
[-18, -17, -15, -14, -5, -1, 0, 2, 4, 7]
```

In [57]:
```
"""
Givem a list of numbers, make another list sorted in non-decreasing order.
"""
import random



# Code - Insertion Sort
def insertion_sort(L):
    for j in range(0, len(L)):
        key = L[j]
        i = j - 1
        while i >= 0 and L[i] > key:
            L[i+1] = L[i]
            i = i - 1
        L[i+1] = key
    return L


# Input
n = 10
L = []
for x in range(n):
    y = random.randint(-20, 30)
    L.append(y)
print(f"Input = {L}")

# Output
print(f"Output = {insertion_sort(L)}")
```

```
Input = [20, 24, -17, 12, 21, 2, 27, 30, 0, -6]
Output = [-17, -6, 0, 2, 12, 20, 21, 24, 27, 30]
```

In [58]:
```
def print_success():
    print("Done")
```

In [59]:
```
print_success()
```

```
Done
```

Doc String

In [60]:
```
# Functions - Doc string
def print_hello():
    """this function prints hello"""
    print("Hello")
```

In [61]:
```
print_hello()
```

```
Hello
```

In [62]:
```
print_hello?
```

In [63]:
```
help(print_hello)
```

```
Help on function print_hello in module __main__:

print_hello()
    this function prints hello
```

More Functions

In [64]:
```
def print_message(msg):
    """ Prints string provided by the user or prints error message."""
    if isinstance(msg, str):
        print(msg)
    else:
        print("input is not a string")
        print("this is the type you supplied", type(msg))
```

In [65]:
```
help(print_message)
```

```
Help on function print_message in module __main__:

print_message(msg)
    Prints string provided by the user or prints error message.
```

In [66]:
```
print_message("ET telephone home")
```

```
ET telephone home
```

In [67]:
```
print_message(3)
```

```
input is not a string
this is the type you supplied <class 'int'>
```

In [68]:
```
print_message??
```

In [69]:
```
def my_add(x, y):
    """ Add two numbers and return the sum """
    return x + y
```

In [70]:
```
my_add(2, 3)
```

Out[70]:
```
5
```

In [71]:
```
def foo():
    x = 1
```

In [72]:
```
foo()
```

In [73]:
```
print(foo())
```

```
None
```

In [74]:
```
print(type(foo()))
```

```
<class 'NoneType'>
```

In [75]:
```
def bar():
    a = 3
    b = 2
    c = 'aloha'
    return a, b, c
```

In [76]:
```
bar()
```

Out[76]:
```
(3, 2, 'aloha')
```

Functions - variable number of input arguments

In [77]:
```
def another_sum(*args):
    sum = 0
    for i in range(len(args)):
        sum += args[i]
    return sum
```

In [78]:
```
another_sum(14, 7, 4, 2)
```

Out[78]:
```
27
```

In [79]:
```
another_sum(14, 7, 4, 2, 1)
```

Out[79]:
```
28
```

In [80]:
```
def print_list(**c):
    for x in c:
        print(x, c[x])
```

In [81]:
```
print_list(c1 = "A", c2 = "B")
```

```
c1 A
c2 B
```

In [82]:
```
print_list(c2 = "A", c5 = "B")
```

```
c2 A
c5 B
```

Functions - Default values

In [83]:
```
def func_default(sum = 0):
    print(sum)
```

In [84]:
```
func_default(24)
```

```
24
```

In [85]:
```
func_default()
```

```
0
```

Modules

import sys

sys.path.append('D:/mymodules/')

import my_functions as foo

foo.my_print('hello')

In [86]:
```
import sys
sys.path.append('D:/Programming/Jupyter/Hello/modules/')
```

In [87]:
```
import my_functions
```

In [88]:
```
my_functions.add_numbers(2, 3, 4)
```

Out[88]:
```
9
```

TODO: invert logic of my_functions.check_not_a_number

In [89]:
```
my_functions.check_not_a_number("hi")
```

Out[89]:
```
False
```

In [90]:
```
from my_functions import check_not_a_number as NaN
```

In [91]:
```
NaN(3)
```

Out[91]:
```
True
```

In [92]:
```
from search_min import search_min_value
```

In [95]:
```
n = 6
A = [5, 1, 7, 3, 10, 11]
```

In [96]:
```
search_min_value(A, n)
```

Out[96]:
```
{'min_value': 1, 'index': 1}
```

String

In [1]:
```
s = "python is the best language for data science"
t = "in this course we are going to learn python"
print(s + 'and' + t)
```

```
python is the best language for data scienceandin this course we are going to learn
python
```

In [3]:
```
price = 12
s = "The price of this book"
v = s + ' is: ' + str(price)
print(v)
```

```
The price of this book is: 12
```

In [7]:
```
multi_line_str = """This is the first line
and here is the second line
and here is the third line
                """
print(multi_line_str)
```

```
This is the first line
and here is the second line
and here is the third line
```

In [8]:
```
print("""The following options are available:
            -a   : does nothing
            -b   : also does nothing
""")
```

```
The following options are available:
            -a   : does nothing
            -b   : also does nothing
```

In [11]:
```
a = "Game of programming"
print(a[3:8])
print(a[-8:-3])
print(len(a))
print(len(a[3:8]))
print(len(a[-8:-3]))
```

```
e of
gramm
19
5
5
```

Strings are immutable

In [13]:
```
a[1] = "i"
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-13-8ea04608f527> in <module>
----> 1 a[1] = "i"

TypeError: 'str' object does not support item assignment
```

In [19]:
```
a[1:12]
```

Out[19]:
```
'ame of prog'
```

In [15]:
```
# a[start:end:step]
a[0:12:2]
```

Out[15]:
```
'Gm fpo'
```

In [16]:
```
# up to ind - 1
ind = 12
a[:ind]
```

Out[16]:
```
'Game of prog'
```

In [17]:
```
print(len(a[:ind]))
```

```
12
```

In [18]:
```
# starts at ind
ind = 3
a[3:]
```

Out[18]:
```
'e of programming'
```

In [20]:
```
# revert string
a[::-1]
```

Out[20]:
```
'gnimmargorp fo emaG'
```

In [25]:
```
a = "     A lot OF Spaces at The    beGinning and end    "
b = a.strip()  # remove leading and trailing spaces
print(b)
print(b.lower())
print(b.upper())
print(a.replace(" ", "-"))
L = "game, and, no".split(",")  # returns list
print(L)
print(b.capitalize())
```

```
A lot OF Spaces at The    beGinning and end
a lot of spaces at the    beginning and end
A LOT OF SPACES AT THE    BEGINNING AND END
-----A-lot-OF-Spaces-at-The----beGinning-and-end----
['game', ' and', ' no']
A lot of spaces at the    beginning and end
```

In [27]:
```
print("at" in "attr")
print("at" not in "attr")
```

```
True
False
```

In [28]:
```
"We are learning \"Strings\" here."
```

Out[28]:
```
'We are learning "Strings" here.'
```

In [29]:
```
'We are learning "Strings" here.'
```

Out[29]:
```
'We are learning "Strings" here.'
```

In [30]:
```
print("c:\drive\name")
print(r"c:\drive\name")
```

```
c:\drive
ame
c:\drive\name
```

In [31]:
```
"abcdefghi" < "def"
```

Out[31]:
```
True
```

In [32]:
```
"abc" == "abc"
```

Out[32]:
```
True
```

In [33]:
```
"$%" < "*&"
```

Out[33]:
```
True
```

In [34]:
```
"$%" > "*&"
```

Out[34]:
```
False
```

Data Structures

List

ordered, changeable, duplicates

In [56]:
```
L = [12, "banana", 5.3]
```

In [57]:
```
L[1]
```

Out[57]:
```
'banana'
```

In [58]:
```
L = L + ["game"]
print(L)
```

```
[12, 'banana', 5.3, 'game']
```

In [59]:
```
L[2] = "orange"
print(L)
```

```
[12, 'banana', 'orange', 'game']
```

In [60]:
```
del L[1]
print(L)
```

```
[12, 'orange', 'game']
```

In [62]:
```
L2 = L
print(L2)
```

```
[12, 'pear', 'game']
```

In [63]:
```
L2[1] = 'pear'
print(L2)
```

```
[12, 'pear', 'game']
```

In [64]:
```
print(L)
```

```
[12, 'pear', 'game']
```

In [68]:
```
# L2 = L makes L2 point to the same address as L
# to copy a list to another memory address use copy()
```

In [65]:
```
L3 = L.copy()
print(L3)
```

```
[12, 'pear', 'game']
```

In [66]:
```
L3[1] = 'grape'
print(L3)
```

```
[12, 'grape', 'game']
```

In [67]:
```
print(L)
```

```
[12, 'pear', 'game']
```

In [20]:
```
del L
```

In [21]:
```
print(L)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-21-70501d4a5d78> in <module>
----> 1 print(L)

NameError: name 'L' is not defined
```

In [69]:
```
L2.append(6.8)
print(L2)
```

```
[12, 'pear', 'game', 6.8]
```

Tuple

ordere, unchangeable, duplicates

In [24]:
```
T = (12, "banana", 5.3)
print(T)
```

```
(12, 'banana', 5.3)
```

In [25]:
```
T[2]
```

Out[25]:
```
5.3
```

In [27]:
```
# Immutable
T[2] = "cherry"
print(T[2])
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-27-d213fc9e6140> in <module>
      1 # Immutable
----> 2 T[2] = "cherry"
      3 print(T[2])

TypeError: 'tuple' object does not support item assignment
```

In [29]:
```
T2 = T
print(T2)
```

```
(12, 'banana', 5.3)
```

In [47]:
```
T[:1]
```

Out[47]:
```
(12,)
```

In [48]:
```
T[::-1]
```

Out[48]:
```
(5.3, 'banana', 12)
```

In [51]:
```
T2 = ('a', 'b', 45)
T3 = T + T2
print(T3)
```

```
(12, 'banana', 5.3, 'a', 'b', 45)
```

Set

unordered, addable/removable, no duplicates

In [70]:
```
S = {12, "banana", 5.3}
print(S)
```

```
{'banana', 12, 5.3}
```

In [71]:
```
12 in S
```

Out[71]:
```
True
```

In [72]:
```
S.add("new item")
print(S)
```

```
{'banana', 'new item', 12, 5.3}
```

In [73]:
```
S.update({"multiple", "items"})
print(S)
```

```
{'new item', 5.3, 'banana', 12, 'multiple', 'items'}
```

In [74]:
```
S.remove("banana")
print(S)
```

```
{'new item', 5.3, 12, 'multiple', 'items'}
```

In [75]:
```
S2 = S.copy()
print(S2)
```

```
{'new item', 5.3, 12, 'multiple', 'items'}
```

In [76]:
```
# does not repeat items
S.add(12)
print(S)
```

```
{'new item', 5.3, 12, 'multiple', 'items'}
```

In [79]:
```
S.add(42)
print(S)
```

```
{'new item', 5.3, 42, 12, 'multiple', 'items'}
```

In [80]:
```
print(S2)
```

```
{'new item', 5.3, 12, 'multiple', 'items'}
```

In [40]:
```
del S[0]
print(S)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-40-fab0d55eae48> in <module>
----> 1 del S[0]
      2 print(S)

TypeError: 'set' object doesn't support item deletion
```

Dictionary

unordered, changeable, no duplicate

In [41]:
```
D = {"id": 12, "name": "Hans"}
print(D)
```

```
{'id': 12, 'name': 'Hans'}
```

In [42]:
```
D['id']
```

Out[42]: 12

In [44]:
```
D['id'] = 20
print(D)
```

```
{'id': 20, 'name': 'Hans'}
```

In [45]:
```
del D['id']
print(D)
```

```
{'name': 'Hans'}
```

In [46]:
```
# include new items
D["new"] = "what"
print(D)
```

```
{'name': 'Hans', 'new': 'what'}
```

In [52]:
```
D2 = {"key1": "morning", "day time": "afternoon"}
```

In [53]:
```
D.update(D2)
```

In [54]:
```
D
```

Out[54]:
```
{'name': 'Hans', 'new': 'what', 'key1': 'morning', 'day time': 'afternoon'}
```

In [84]:
```
L = [7, 'brazil', 5.8, 'useful']
```

In [82]:
```
T = (4, 6, 2, 'hi')
```

In [83]:
```
S = {'profile', 89, 'key', 5.0}
```

In [85]:
```
D = {'num': 34, 'user': 'david', 'credit': 'Visa'}
```

In [86]:
```
D2 = {'A': L, 'B': T, 'C': S, 'D': D}
```

In [87]:
```
D2
```

Out[87]:
```
{'A': [7, 'brazil', 5.8, 'useful'],
 'B': (4, 6, 2, 'hi'),
 'C': {5.0, 89, 'key', 'profile'},
 'D': {'num': 34, 'user': 'david', 'credit': 'Visa'}}
```

In [88]:
```
D2['A']
```

Out[88]:
```
[7, 'brazil', 5.8, 'useful']
```

In [89]:
```
D2['A'][2]
```

Out[89]:
```
5.8
```

In [90]:
```
K = D2['D']
```

In [91]:
```
K
```

Out[91]:
```
{'num': 34, 'user': 'david', 'credit': 'Visa'}
```

In [92]:
```
for x in K:
    print(x, K[x])
```

```
num 34
user david
credit Visa
```

In [93]:
```
L3 = [L, T, D, 23, 'game']
```

In [94]:
```
type(L3[2])
```

Out[94]:
```
dict
```

In [95]:
```
L3 = [x**2 for x in range(10)]
```

In [96]:
```
L3
```

Out[96]:
```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

In [97]:
```
S3 = {x**2 for x in range(2, 20, 3)}
```

In [98]:
```
S3
```

Out[98]:
```
{4, 25, 64, 121, 196, 289}
```

In [99]:
```
"""Let say you are a teacher and you have diferente students records containing id of a student and the marks list in each subject where different students have taken different number of subjects. All these records are in hard copy. You want to enter all the data in computer and want to compute the average marks of each student and display."""
```

Out[99]:
```
'Let say you are a teacher and you have diferente students records containing id of a student and the marks list in each subject where different students have taken different number of subjects. All these records are in hard copy. You want to enter all the data in computer and want to compute the average marks of each student and display.'
```

In [100]:
```
def get_user_data():
    D = {}
    while True:
        student_id = input("Enter student ID: ")
        marks_list = input("Enter the marks by comma separated values: ")
        more_students = input("Enter yes/no for adding more students: ")
        if student_id in D:
            print(student_id, "is already inserted.")
        else:
            D[student_id] = marks_list.split(",")
        if more_students.lower() == "no":
            return D
```

In [101]:
```
student_data = get_user_data()
```

```
Enter student ID: 4
Enter the marks by comma separated values: 7, 8.6, 9, 10
Enter yes/no for adding more students: yes
Enter student ID: 7
Enter the marks by comma separated values: 3, 10, 9, 8
Enter yes/no for adding more students: yes
Enter student ID: 1
Enter the marks by comma separated values: 5, 4
Enter yes/no for adding more students: no
```

In [102]:
```
student_data
```

Out[102]:
```
{'4': ['7', ' 8.6', ' 9', ' 10'],
 '7': ['3', ' 10', ' 9', ' 8'],
 '1': ['5', ' 4']}
```

In [115]:
```
def get_average(D):
    avg_mark = {}
    for x in D:
        L = D[x]
        s = 0
        for marks in L:
            s += int(float(marks))
        avg_mark[x] = s/len(L)
    return avg_mark
```

In [116]:
```
avg_marks = get_average(student_data)
print(avg_marks)
```

```
{'4': 8.5, '7': 7.5, '1': 4.5}
```

In [117]:
```
for x in avg_marks:
    print("Student ", x, "got average marks ", avg_marks[x])
```

```
Student  4 got average marks  8.5
Student  7 got average marks  7.5
Student  1 got average marks  4.5
```

In [ ]: