

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`

In []: