Explaining Markdown Cells

Header 2

Header 3

- Header 4

Header 5

Header 6

In [29]:
```
#Coding cell blocks
```

In [1]:
```
print("Hello Farah")
```

```
Hello Farah
```

Input function ( )

- It prompts the user for input, waits for the user to type a line of text, and then returns that text as a string.

In [1]:
```
input("what is your name")
```

```
what is your nameFarah
```

Out[1]:
```
'Farah '
```

In [2]:
```
a = input("what is your name?")
```

```
what is your name?Farah
```

In [3]:
```
print(a)
```

```
Farah
```

input("what is your name") a = input("what is your name?")

The first example displays a prompt but does not store the input, while the second example displays a prompt and stores the user's input in the variable `a`.

In [4]:
```
"Hello " + input("what is your name") + ",  How are you?"
```

```
what is your nameFarah
```

Out[4]:
```
'Hello Farah,  How are you?'
```

The above is an example of string concatenation in Python, where the input from the user is combined with other strings to form a single output string.

In [5]:
```
name = 'Farah'
print (name)
```

```
Farah
```

```
greeting = 'Hello'
```

In [16]:
```
print(greeting)
```

```
Hello
```

In [30]:
```
50+50
```

Out[30]:
```
100
```

In [31]:
```
"50" + "50"
```

Out[31]:
```
'5050'
```

In [3]:
```
x = 10
y = 5

add = x+y

print("Addition value is:", add)
```

```
Addition value is: 15
```

- The code adds two variables, `x` and `y`, and prints the result with a message.

In [6]:
```
a = 450
b = -625

add = a+b
Multiply = a*b
equation = (a*b)/(a-b)

print ("the final answer is:", add)
print ("the final answer is:", Multiply)
print ("the final answer is:", equation)
```

```
the final answer is: -175
the final answer is: -281250
the final answer is: -261.6279069767442
```

In [7]:
```
a = 450
b = -625
c = 12.78

Subtract = a-b-c
Divide = a+b/c
equation = (a*b+c)/(a-c)

print ("the final answer is:", Subtract)
print ("the final answer is:", Divide)
print ("the final answer is:", equation)
```

```
the final answer is: 1062.22
the final answer is: 401.09546165884194
the final answer is: -643.2396047756278
```

In [8]:
```
line1 = 'my name is farah'
line1
```

Out[8]:
```
'my name is farah'
```

In [9]:
```
line1 = "hello, my name is Farah Ibrar"
print(line1.capitalize())
```

```
Hello, my name is farah ibrar
```

- The code capitalizes the first letter of the string stored in line1 and prints the modified string.

In [10]:
```
example = "my name is umar"
print (example.capitalize())
```

```
My name is umar
```

In [11]:
```
line2 = "I am from pakistan"
print(line2.upper())
```

```
I AM FROM PAKISTAN
```

- Converts all letters in the string line2 to uppercase and prints the result.

In [33]:
```
line2 = "I am from pakistan"
print(line2.title())
```

```
I Am From Pakistan
```

- This converts the first letter of each word in the string line2 to uppercase and all other letters to lowercase, and then prints the result.

In [36]:
```
line2 = "I AM from Pakistan"
print(line2.lower())
```

```
i am from pakistan
```

- This converts all letters in the string line2 to lowercase and prints the result.

In [37]:
```
x = True
y = False

print (x,y)
```

```
True False
```

In [12]:
```
line2 = "I am from pakistan"
x = "Yes"

print (line2,x)
```

```
I am from pakistan Yes
```

In [13]:
```
a = True
b = False

x = 20
y = 55


z = int(input("Enter a value:"))
if z >= 50:
    print(a)
else:
    print(b)
```

```
Enter a value:599
True
```

This code prompts the user to enter a value, converts the input to an integer, and then checks if the entered value (z) is greater than or equal to 50. Depending on the result:

- If z is greater than or equal to 50, it prints True (a is True).
- If z is less than 50, it prints False (b is False).

In [14]:
```
num_1 = int(input("Enter a value:"))
num_2 = int(input("Enter a value:"))

num_1 == num_2
```

```
Enter a value:10
Enter a value:50
```
Out[14]:
```
False
```

This above line of code checks if num_1 is equal to num_2. It evaluates to either True if they are equal or False if they are not equal.

In [15]:
```
a = True
b = False

x = 20
y = 55

z = int(input("Enter a value:"))
if x and y >= z :
    print(a)
else:
    print(b)
```

```
Enter a value:32
True
```

The code prompts the user to enter a value, converts it to an integer, and then checks if both x and y are greater than or equal to the entered value z. Depending on the result:

- If both x and y are greater than or equal to z, it prints True (a is True).
- Otherwise, it prints False (b is False).

In [23]:
```
type(int(input("Enter a value:")))
```

```
Enter a value:11
```
Out[23]:
```
int
```

This line of code prompts the user to enter a value, converts the input to an integer using int(), and then determines the type of the resulting value using the type( ) function.

The `type(int(input("Enter a value:")))` expression can check the type of any input that can be successfully converted to an integer using int(). This includes numeric values (integers and floats) as well as strings that represent valid integer literals. Here are a few examples:

- If you enter `"123"`, it will output `<class 'int'>`.
- If you enter `123`, it will output `<class 'int'>`.
- If you enter `12.34`, it will output `<class 'int'>` after converting it to `12`.

When you enter `12.34`, int( ) cannot convert it directly to an integer because int( ) expects a whole number or a string representation of a whole number.

In Python, you can only convert a string to an integer directly if it represents a valid integer literal.

To handle decimal input like `12.34`, you would need to use `float( )` instead of `int( )` to convert it to a floating-point number, and then you can check its type. Here's how you can modify the code to handle both integer and float input correctly:

In [25]:
```
value = input("Enter a value:")
try:
    num = int(value)
except ValueError:
    try:
        num = float(value)
    except ValueError:
        print("Invalid input. Please enter a valid number.")
        num = None

if num is not None:
    print(type(num))
```

```
Enter a value:22
<class 'int'>
```

This code:

1. Prompts the user to enter a value.
2. Attempts to convert the input to an integer using `int( )`.
3. If that fails (because the input contains a decimal point or is not a number at all), it attempts to convert the input to a float using `float( )`.
4. If both conversion attempts fail, it prints an error message.
5. If the conversion is successful, it prints the type of the resulting number.