

Today's Agenda

- Conditional statements
- Nested conditional statements
- Loops
- Console based game development

Conditional Statements

Definition:

- Conditional statements help a computer to make decision based on certain conditions.
- A programmer writes a conditional statements in order to make a computer to perform an action if certain criteria meets its requirement or happens to be true.

if - else

```
if ():  
    # do addition  
else:  
    # do subtraction
```

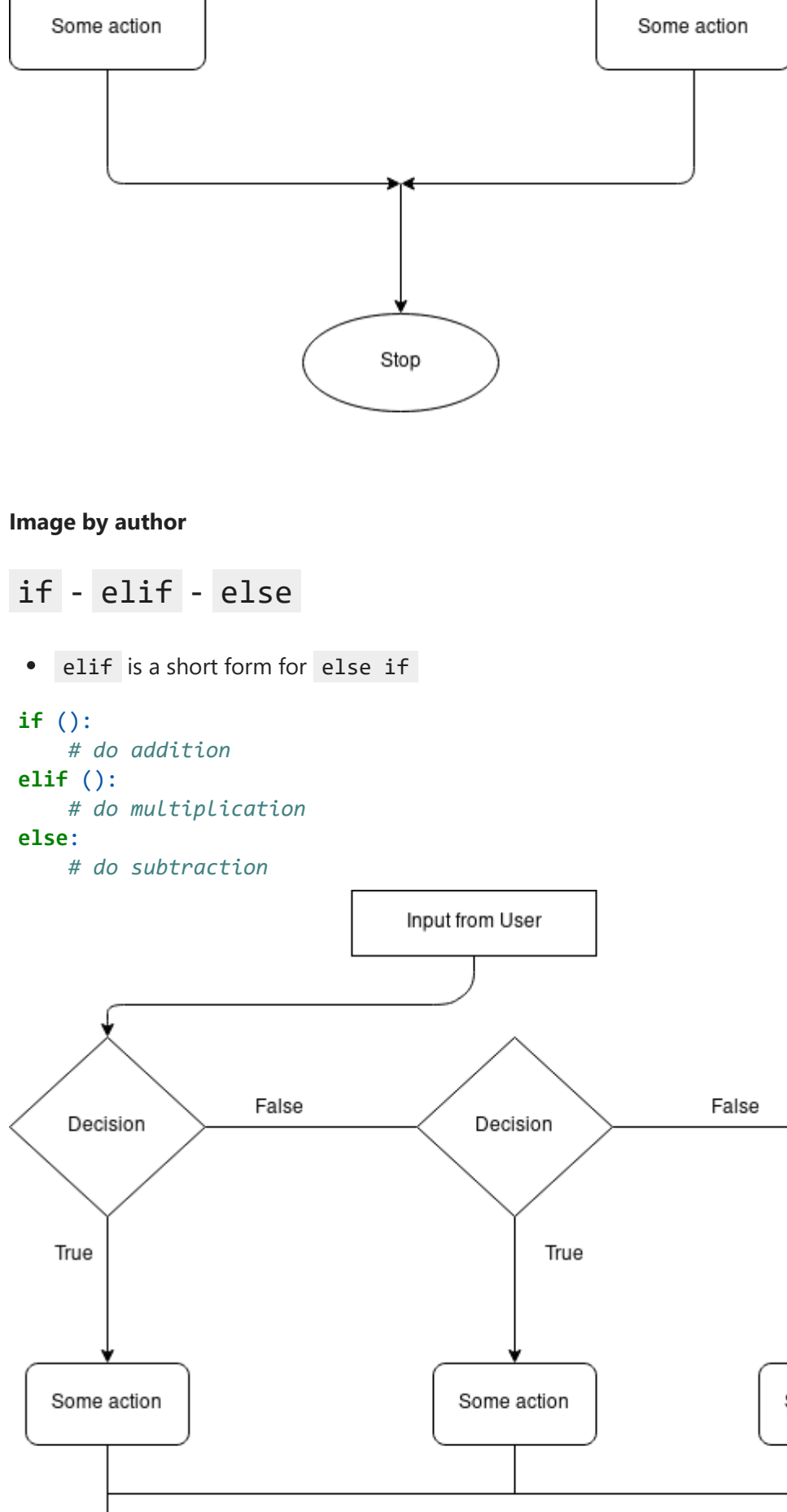


Image by author

if - elif - else

- elif is a short form for else if

```
if ():  
    # do addition  
elif ():  
    # do multiplication  
else:  
    # do subtraction
```

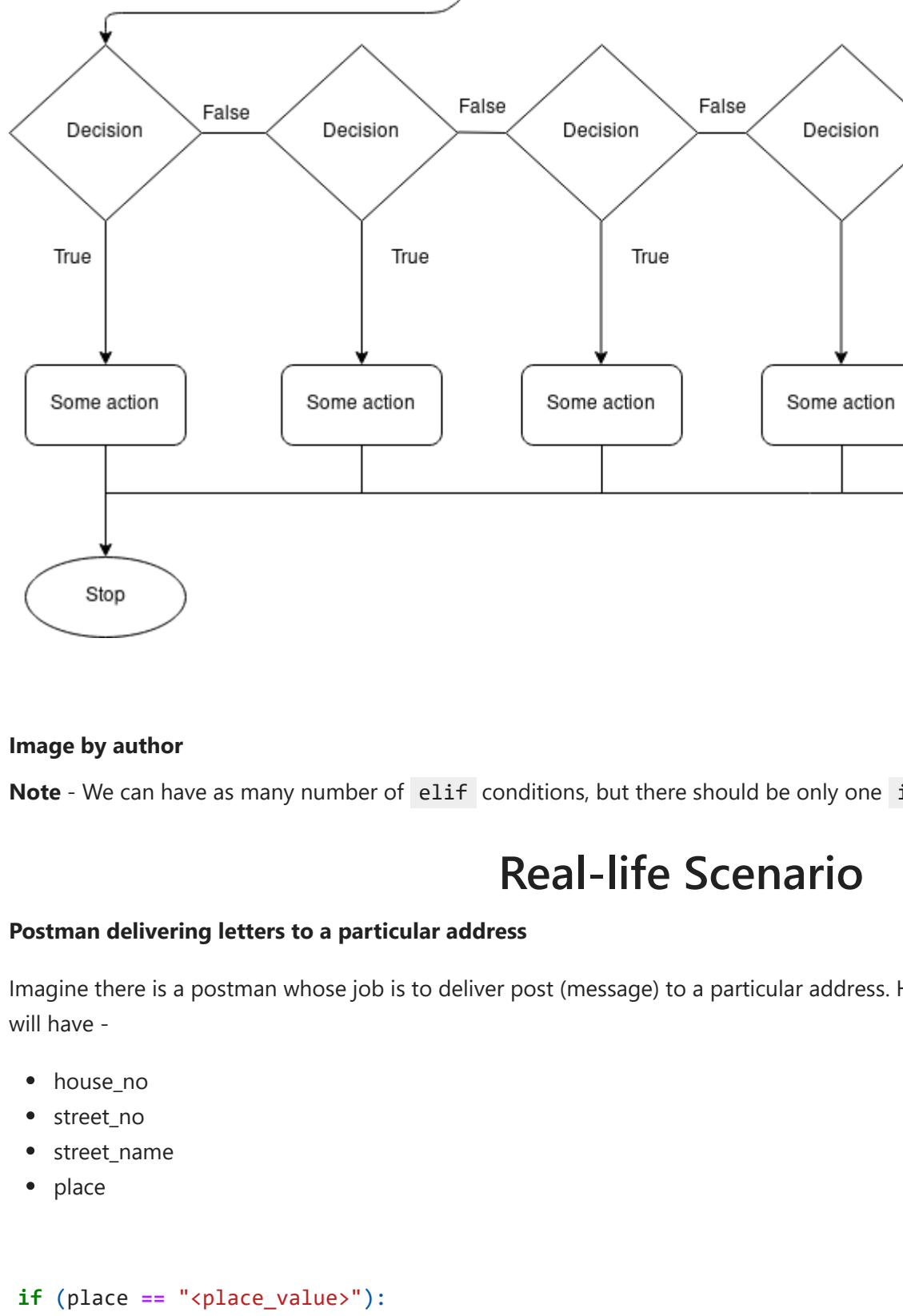


Image by author

if - elif - ... - elif - else

```
if ():  
    # do addition  
elif ():  
    # do multiplication  
elif ():  
    # do something  
elif ():  
    # do something  
else:  
    # do subtraction
```

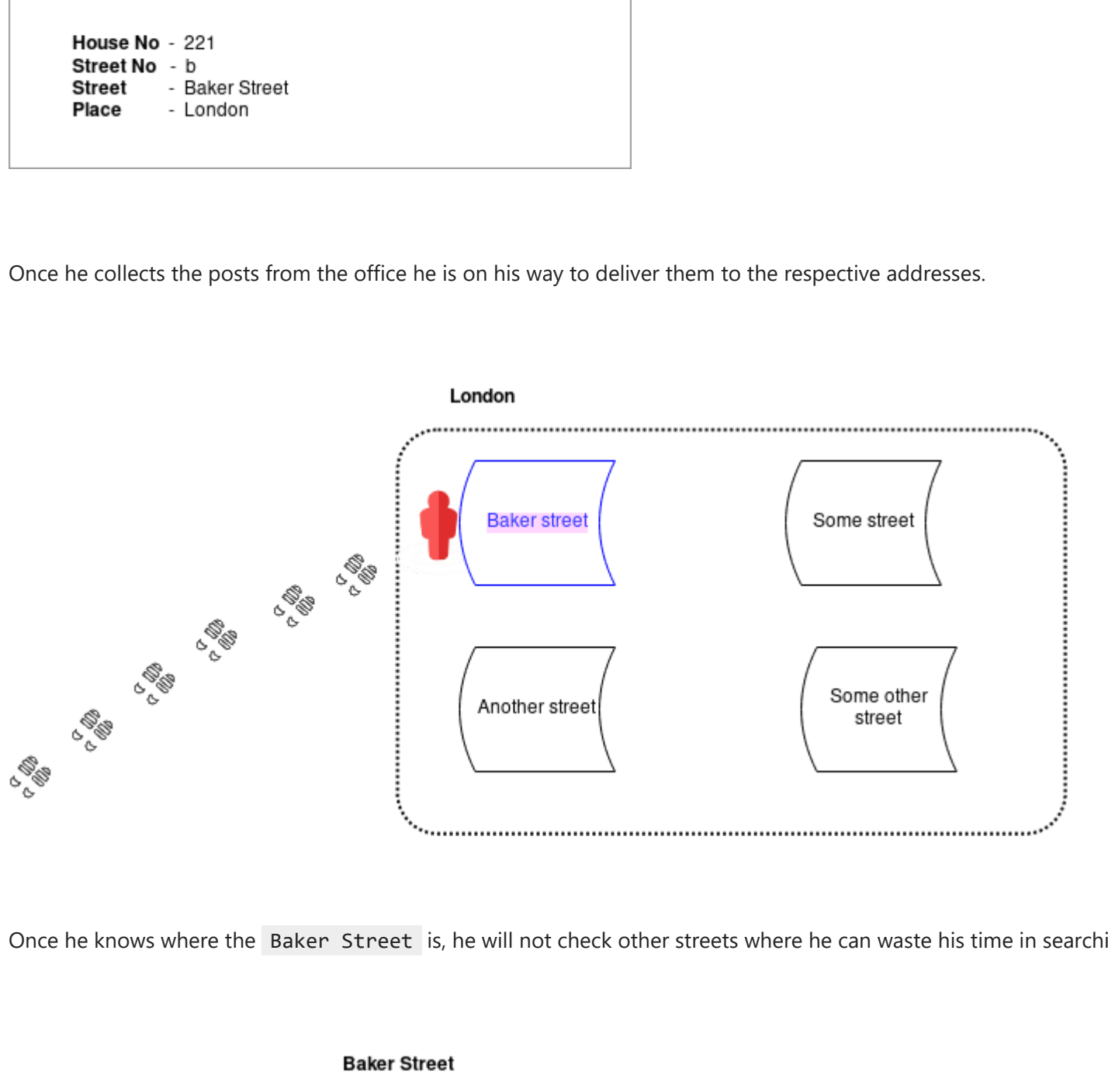


Image by author

Note - We can have as many number of elif conditions, but there should be only one if and one else if following this chain.

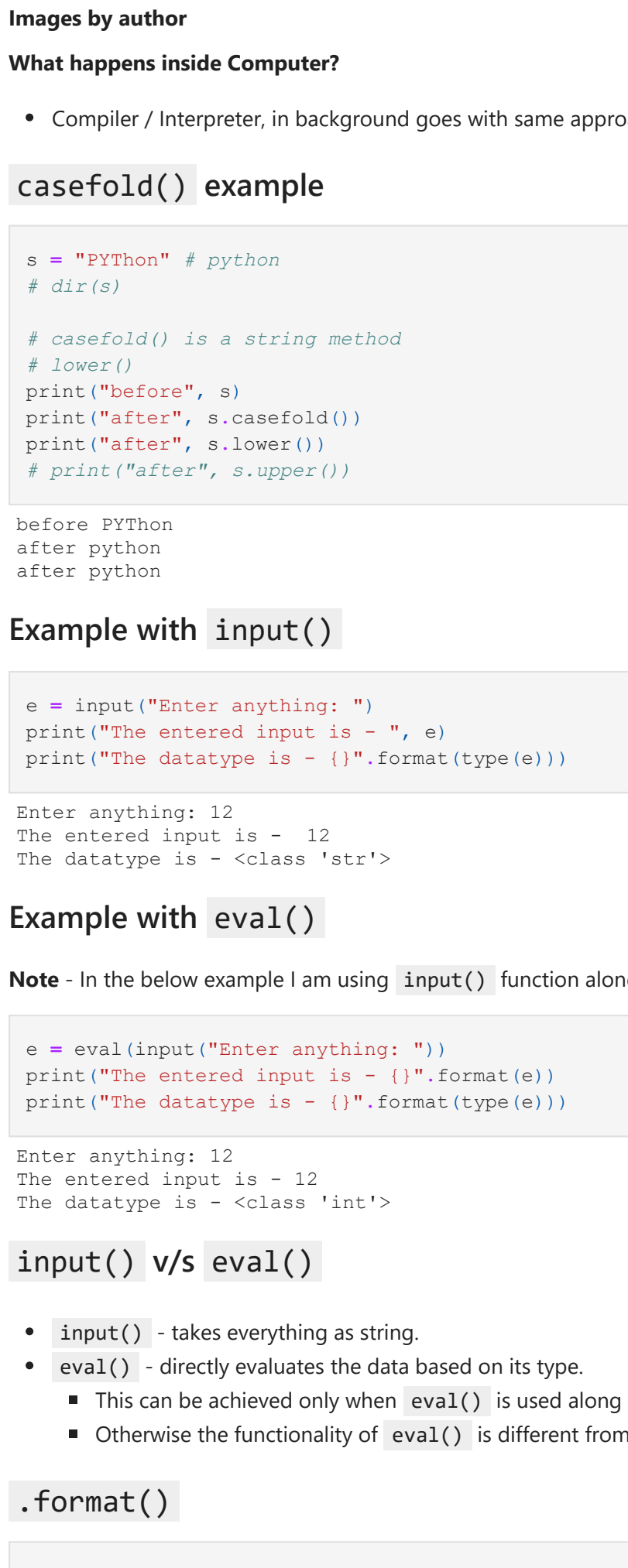
Real-life Scenario

Postman delivering letters to a particular address

Imagine there is a postman whose job is to deliver post (message) to a particular address. He is given only a specific address in which he will have -

- house_no
- street_no
- street_name
- place

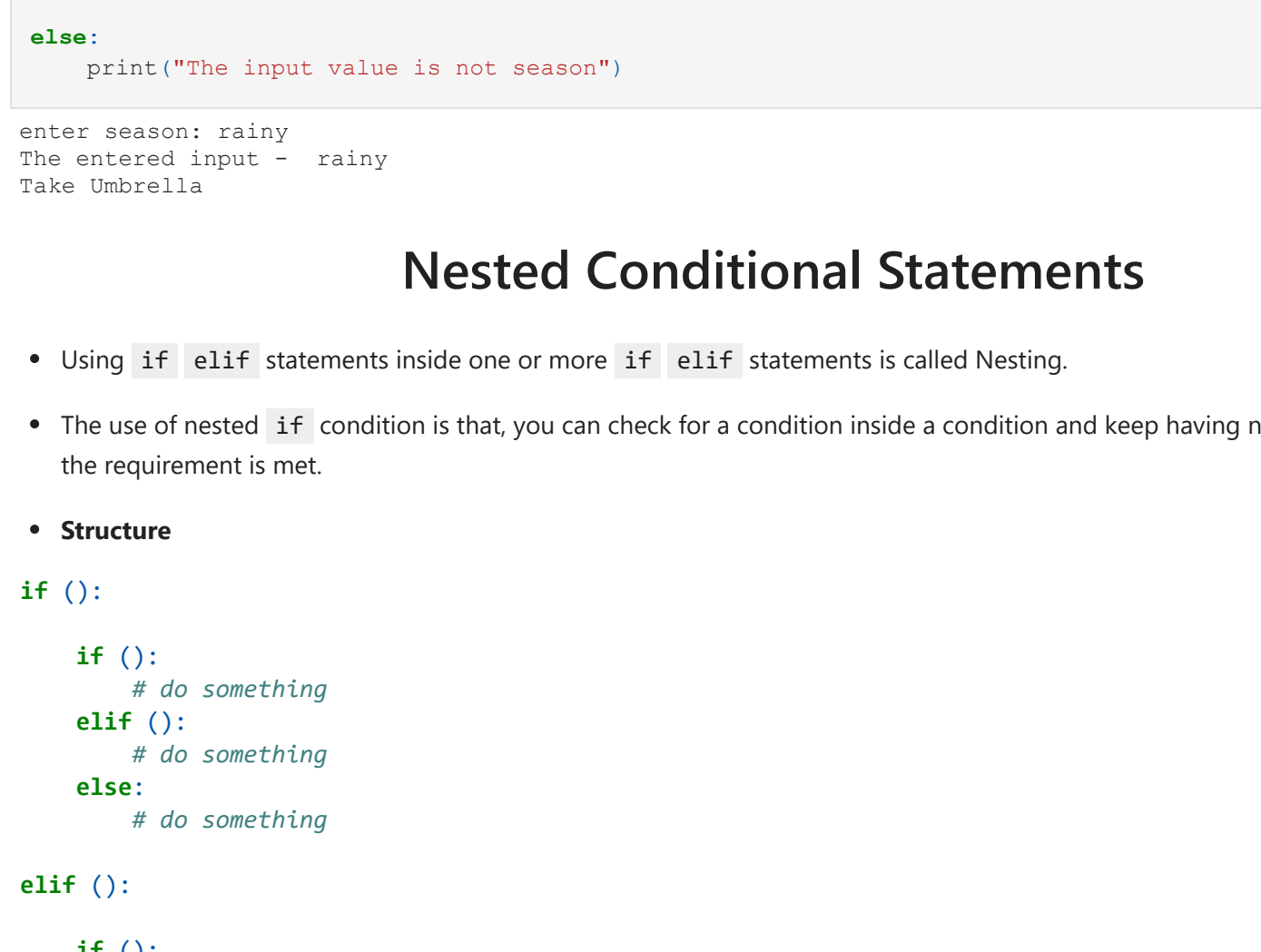
```
if (place == "<place_value>"):  
    if (street_name == "<st_name_value>"):  
        if (street_no == "<st_no_value>"):  
            if (house_no == "<house_no_value>"):  
                print("Post is successfully delivered")  
            else:  
                print("House no - invalid")  
        else:  
            print("Street no - invalid")  
    else:  
        print("Street name - invalid")  
else:  
    print("Place - invalid")
```



Once he collects the posts from the office he is on his way to deliver them to the respective addresses.



Once he knows where the 'Baker Street' is, he will not check other streets where he can waste his time in searching.



Once he knows where the street number 'B' is, he will not check other street numbers where he can waste his time in searching.



Once he knows where the respective door number '221' is, he will not check other door numbers where he can waste his time in searching.

Images by author

What happens inside Computer?

- Compiler / Interpreter, in background goes with same approach in discarding other conditions once the required condition is met.

casefold() example

```
In [1]: s = "PYthon" # python  
# dir(s)  
# casefold() is a string method  
# lower()  
print("before", s)  
print("after", s.casefold())  
print("after", s.lower())  
# print("after", s.upper())
```

before PYthon
after python
after python

Example with input()

```
In [2]: e = input("Enter anything: ")  
print("The entered input is - {}".format(e))  
print("The datatype is - {}".format(type(e)))
```

Enter anything: 12
The entered input is - 12
The datatype is - <class 'str'>

Example with eval()

Note - In the below example I am using input() function along with eval().

```
In [3]: e = eval(input("Enter anything: "))  
print("The entered input is - {}".format(e))  
print("The datatype is - {}".format(type(e)))
```

Enter anything: 12
The entered input is - 12
The datatype is - <class 'int'>

input() v/s eval()

- input() - takes everything as string.
- eval() - directly evaluates the data based on its type.
 - This can be achieved only when eval() is used along with input().
 - Otherwise the functionality of eval() is different from input().

.format()

```
In [4]: 'hi ' + str(12)
```

Out[4]: 'hi 12'

```
In [5]: # Python 1 - 2  
print("Python 1 - 2")  
s0 = 'Python '  
s1 = 1  
s2 = 2  
s = s0 + str(s1) + ' - ' + str(s2)  
print(s)  
s = "{}() - {}".format(s0, s1, s2)  
print(s)
```

Python 1 - 2
Python 1 - 2
Python 1 - 2

Example of if, elif and else

```
In [6]: in_ = input("enter season: ")  
print("The entered input - ", in_)  
if (in_ == "rainy"):  
    print("Take Umbrella")  
elif (in_ == "winter"):  
    print("Wear Sweater")  
elif (in_ == "summer"):  
    print("Consume Watery (Healthy) Food")  
elif (in_ == "spring"):  
    print("Do Exercise")  
else:  
    print("The input value is not season")
```

enter season: rainy
The entered input - rainy
Take Umbrella

Nested Conditional Statements

- Using if elif statements inside one or more if elif statements is called Nesting.
 - The use of nested if condition is that, you can check for a condition inside a condition and keep having nested conditions until the requirement is met.
- **Structure**

```
if ():  
    if ():  
        # do something  
    elif ():  
        # do something  
    else:  
        # do something  
elif ():  
    if ():  
        # do something  
    elif ():  
        # do something  
    else:  
        # do something  
else:  
    if ():  
        # do something  
    elif ():  
        # do something  
    else:  
        # do something
```

Note - I will leave the flow chart representation to you itself. Most of it is same and can be drawn easily.

Example of nested if conditions

```
In [7]: in_ = eval(input("enter season: "))  
print(in_)  
if isinstance(in_, str):  
    print(type(in_))  
    in_ = in_.casefold()  
    if (in_ == "rainy"):  
        print("Take Umbrella")  
    elif (in_ == "winter"):  
        print("Wear Sweater")  
    elif (in_ == "summer"):  
        print("Consume Watery (Healthy) Food")  
    elif (in_ == "spring"):  
        print("Do Exercise")  
    else:  
        print("The input value is not season")  
else:  
    print(type(in_))  
    print("Input not understood!!!")
```

enter season: 12
12
<class 'int'>
Input not understood!!!

Loops in Python

Loop is basically a sequence of instructions that is repeated continually and checked until a certain condition is met or happens to be True.

Types of loops

- While loop
- for loop
- Nested loop - can be of both 'for' and 'while'

print("hi") 10 times using while

```
In [8]: counter = 1  
while (counter <= 11):  
    print('hi --> ', counter)  
    counter = counter + 1  
else:  
    print("Bye")
```

hi --> 1
hi --> 2
hi --> 3
hi --> 4
hi --> 5
hi --> 6
hi --> 7
hi --> 8
hi --> 9
hi --> 10
Bye

```
In [9]: # import time  
# counter = 1  
# while (True):  
#     print('hi --> ', hi_.format(counter))  
#     time.sleep(1)  
#     counter = counter + 1
```

print("hi") 10 times using for

Note: For the working of for loop, we need a 'sequence object' to iterate through each element and perform the execution.

- range() gives a sequence object.
- We can also have a string / list / array / tuple / dictionary as a sequence object.

```
In [10]: # help(range)  
list(range(1, 11))
```

Out[10]: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

```
In [11]: for counter in range(1, 11):  
    print('hi --> ', counter)
```

hi --> 1
hi --> 2
hi --> 3
hi --> 4
hi --> 5
hi --> 6
hi --> 7
hi --> 8
hi --> 9
hi --> 10

Elementwise adding of two lists using while loop

```
In [12]: l1 = [1, 2, 3, 4, 5]  
l2 = [4, 5, 10, 12, 14]  
print(len(l1))  
print(len(l2))
```

5
5

```
In [13]: l1 + l2 # [5, 7, 13, 16, 19]
```

Out[13]: [1, 2, 3, 4, 5, 4, 5, 4, 5, 10, 12, 14]

```
In [14]: l3 = []  
c = 0  
if (len(l1) == len(l2)):  
    while (c < len(l2)):  
        res = l1[c] + l2[c]  
        l3.append(res)  
        c += 1  
else:  
    print("Lengths do not match")
```

```
In [15]: print(l3)
```

[5, 7, 13, 16, 19]

Elementwise adding of two lists using for loop

```
In [16]: l1 = [1, 2, 3, 4, 5]  
l2 = [4, 5, 10, 12, 14]  
print(len(l1))  
print(len(l2))
```

5
5

```
In [17]: l3 = []  
if (len(l1) == len(l2)):  
    for i in range(len(l1)):  
        res = l1[i] + l2[i]  
        l3.append(res)  
else:  
    print("Lengths do not match")
```

```
In [18]: print(l3)
```

[5, 7, 13, 16, 19]

Magic code

```
In [19]: l1 = [1, 2, 3, 4, 5]  
l2 = [4, 5, 10, 12, 14]  
print(len(l1))  
print(len(l2))
```

5
5

```
In [20]: if (len(l1) == len(l2)):  
    l3 = list(map(sum, list(zip(l1, l2))))  
else:  
    l3 = []  
    print("Lengths do not match")
```

```
In [21]: print(l3)
```

[5, 7, 13, 16, 19]

Rock Paper Scissors

Console based game development

```
In [22]: ## Let's develop a game using the above concepts ##  
while (True):  
    one_user = input("Sherlock\t: ").casefold()  
    two_user = input("Mycroft\t: ").casefold()  
    if (one_user == two_user):  
        print("Match Draw")  
    elif (one_user == 'r') and (two_user == 'p'):  
        print("Mycroft won the game")  
    elif (one_user == 'r') and (two_user == 's'):  
        print("Sherlock won the game")  
    elif (one_user == 'p') and (two_user == 'r'):  
        print("Sherlock won the game")  
    elif (one_user == 'p') and (two_user == 's'):  
        print("Mycroft won the game")  
    elif (one_user == 's') and (two_user == 'r'):  
        print("Mycroft won the game")  
    elif (one_user == 's') and (two_user == 'p'):  
        print("Sherlock won the game")  
    else:  
        print("Inputs do not match.")  
    print("\n")  
    continue_input = input("Want to play again? (y/n): ").casefold()  
    print("\n")  
    if (continue_input == 'y') or (continue_input == 'yes'):  
        continue  
    else:  
        print("See you again. Bye")  
        break  
Sherlock : r  
Mycroft : r  
Match Draw  
Want to play again? (y/n): n  
See you again. Bye
```

Homework / Exercise

- Take the rock paper scissors code and instead of having 2 humans (users) giving the input, change it to 1 human versus 1 computer.
 - 1 human - should give input as usual.
 - 1 computer - should select random element from r, p, s.
- Hint** -
 - You should use random module which you will need to import.
 - Explore what random module is and how to use it for this example.

What did we learn?

- Conditional Statements
- Nested Conditional Statements
- Loops
 - while loop
 - for loop
- Elementwise adding of two lists
- Control statements break and continue