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 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 codition is met.

`casefold()` example

In [1]:

```
s = "PYThon"
# dir(s)

# casefold() is a string method
# lower()
print("before", s)
print("after", s.casefold())
```

```
before PYThon
after python
```

Example with `input()`

In [2]:

```
e = input("Enter anything: ")
print("The entered input is - ", 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: 123
The entered input is - 123
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()`.

Example of `if`, `elif` and `else`

In [4]:

```
in_ = input("enter season: ").casefold()
print("The entered input - ", in_)

if (in_ == 'rainy'):
    print("Take Umbrella")

elif (in_ == 'winter'):
    print("Wear Sweater")

elif (in_ == 'summer'):
    print("Consume Cool Drinks")

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.

Typical nested conditional statements

```
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 [5]:

```
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 Cool Drinks")

    elif (in_ == 'spring'):
        print("Do Exercise")

    else:
        print("The input value is not season")

else:
    print(type(in_))
    print("Input not understood!!!")
```

```
enter season: "SummEr"
SummEr
<class 'str'>
Consume Cool Drinks
```

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 [6]:

```
counter = 1
while (counter < 11):
    print('{} --> hi'.format(counter))
    counter = counter + 1
```

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

In [7]:

```
# import time

# counter = 1
# while (True):
#     print('{} --> 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 [8]:

```
# help(range)
```

In [9]:

```
print(list(range(0, 11)))
print(list(range(1, 11, 2)))
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
[1, 3, 5, 7, 9]
```

In [10]:

```
for counter in range(1, 11):
    print('{} --> hi'.format(counter))
```

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

Sum of `n` numbers using `while`

In [11]:

```
list(range(1, 11))
```

Out[11]:

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

In [12]:

```
sum(range(1, 11))
```

Out[12]:

```
55
```

In [13]:

```
add_number = 0
counter = 1
n_limit = int(input("Enter n: "))
print("The entered number n is - {}".format(n_limit))

while (counter < n_limit + 1):
    holder = '({} + {})'.format(add_number, counter)
    add_number = add_number + counter
    print(counter, '\t>> ', holder, '\t>> ', add_number)
    counter = counter + 1
```

```
Enter n: 5
The entered number n is - 5
1        >>  (0 + 1)     >>  1
2        >>  (1 + 2)     >>  3
3        >>  (3 + 3)     >>  6
4        >>  (6 + 4)     >>  10
5        >>  (10 + 5)    >>  15
```

Sum of `n` numbers using `for`

In [14]:

```
add_number = 0
n_limit = int(input("Enter n: "))
print("The entered number n is - {}".format(n_limit))

for counter in range(1, n_limit + 1):
    holder = '({} + {})'.format(add_number, counter)
    add_number = add_number + counter
    print(counter, '\t>> ', holder, '\t>> ', add_number)
```

```
Enter n: 5
The entered number n is - 5
1        >>  (0 + 1)     >>  1
2        >>  (1 + 2)     >>  3
3        >>  (3 + 3)     >>  6
4        >>  (6 + 4)     >>  10
5        >>  (10 + 5)    >>  15
```

`format()` method

In [15]:

```
var = "sameer"
occ = "python teaching"

print("hey " + var + " ! what's up? I do " + occ + " - haha")

print("hey {} ! what's up? I do {} - haha".format(var, occ))
```

```
hey sameer ! what's up? I do python teaching - haha
hey sameer ! what's up? I do python teaching - haha
```

Console based game development

Rock Paper Scissors

In [16]:

```
## 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 == 'r') and (two_user == 'r'):
        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 == 'p'):
        print("Match Draw")
    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")
    elif (one_user == 's') and (two_user == 's'):
        print("Match Draw")

    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         : p
Mycroft won the game


Want to play again? (y/n): y


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 it for using it.
- Explore what is `random` module and how to use it for this example.

What did we learn?

- Conditional Statements
- Nested Conditional Statements
- Loops
  - while loop
  - for loop
- Logic for sum of n numbers
- Control statements `break` and `continue`
- Simple game development