Session 4

- If statement
- If else statements
- If elif else statements
- Built in list methods
- Built in String methods
- Mutable vs Immutable datatypes

1) If statement

If statements allows programming languages to make decisions

We can have individual if statements without an else part. The single If statements are useful if we want to check for conditions and do certain thing only if the codition is True.

In [2]:
```
if True:
    print("I am True")
```

```
I am True
```

Usualy the conditions are created using the comparison and logical operators

In [3]:
```
a = 10
b = 20
a<b
```

Out[3]:
```
True
```

In [4]:
```
if a<b:
    print("a is lesser than b")
```

```
a is lesser than b
```

What happens if the condition we provided in the if statement is not true

In [5]:
```
if a>b:
    print("a is lesser than b")
```

We print nothing because the block of code inside the if statement (the indented part is not executed).
This bring us to our next use condition flow statement **If Else**

2) If else statement

We can have `if else blocks` if we want some operation to be done if the condition in the if returns false

In [6]:
```
a = 10
b = 20
if a<b:
    print("a is lesser than b")
else:
    print("b is lesser than a")
```

```
a is lesser than b
```

In [7]:
```
a = 100
b = 20
if a<b:
    print("a is lesser than b")
else:
    print("b is lesser than a")
```

```
b is lesser than a
```

The above code is going to return wrong result if a is equal to b. So we need have a way of testing multiple conditions in side the same if block

In [8]:
```
a = 20
b = 20
if a<b:
    print("a is lesser than b")
else:
    print("b is lesser than a")
```

```
b is lesser than a
```

3) If elif else statement

We can have `if elif else blocks` are used when we want to check for multiple conditions

In [9]:
```
a = 20
b = 20
if a<b:
    print("a is lesser than b")
elif a>b:
    print("b is lesser than a")
else :
    print("both are equal")
```

```
both are equal
```

TASK 1

```
A school has following rules for grading system:
        Below 25 - F
        25 to 45 - E
        45 to 50 - D
        50 to 60 - C
        60 to 80 - B
        Above 80 - A
Ask user to enter marks and print the corresponding grade.
```

In [10]:

```python
a = int(input("enter your marks : "))

if a<25:
    print("F")
elif a>=25 and a<45:
    print("E")
elif a>=45 and a<50:
    print("D")
elif a>=50 and a<60:
    print("C")
elif a>=60 and a<80:
    print("B")
elif a>=80 and a<=100:
    print("A")
else:
    print("not valid")
```

```
enter your marks : 50
C
```

4) Built-in List methods

All datatypes including list are an object in python. Which means we can call methods on the objects created using these classes.
Try creating a list then write the list name followed by a `.` and press the tab key

In [ ]:

```python
my_list = [11,22,33,44,55,66,77,88,99]
my_list.
```

This Gives us a dropdown of all the available built-in list methods we will be looking at a few important ones.

The methods can be :

- In place (affects the actual list)
- not in place (the actual list is not affected)

depending on the type we might have to reassign the variable to see the changes.

The `append()` method adds an element to the end of the list
The `append()` method is an inplace method.

In [34]:

```python
my_list = [11,22,33,44,55,66,77,88,99]
my_list.append(101)
```

In [35]:

```
my_list
```

Out[35]:

```
[11, 22, 33, 44, 55, 66, 77, 88, 99, 101]
```

The `insert()` method adds an element at the specified index
The `append()` method is an inplace method.

In [33]:

```
my_list = [11,22,33,44,55,66,77,88,99]
my_list.insert(4,"I am added before index 4")
print(my_list)
```

```
[11, 22, 33, 44, 'I am added before index 4', 55, 66, 77, 88, 99]
```

5) Built-in String methods

All datatypes including Strings are an object in python. Which means we can call methods on the objects created using these classes.
Try creating a string then write the variable name followed by a `.` and press the tab key

In [ ]:

```
my_string = "hello world"
my_string.
```

The `upper()` method capitalises all characters in the string
The `upper()` method is an not in place method.

In [37]:

```
my_string = "hello world"
print(my_string.upper())
print(my_string)
```

```
HELLO WORLD
hello world
```

Since it is a not in place method we will have to reasign it to the same variable to reflect the changes

In [38]:

```
my_string = "hello world"
my_string = my_string.upper()
print(my_string)
```

```
HELLO WORLD
```

The `split()` method split the strings at a a specified character. By default the split character is space `` `<br> ``
The split() `converts the splited words into a list`
The split()` method is an not in place method.

In [40]:

```
my_string = "hello world How are you"
x = my_string.split()
print(x)
print(my_string)
```

```
['hello', 'world', 'How', 'are', 'you']
hello world How are you
```

6) Mutable and Imutable datatypes

All the collection datatypes in python can be classified in to 2 types.

- mutable (Inidvdual elemnts can be reassigned)
- imutable (Indivisual elements cannot be reassigned)

- **Lists are mutable**
- **Strings are imutable**

In [2]:
```
my_list = [11,22,33,44,55,66]
print(my_list)
my_list[3] = 99
print(my_list)
```

```
[11, 22, 33, 44, 55, 66]
[11, 22, 33, 99, 55, 66]
```

In [3]:
```
my_string = "hello"
print(my_string)
my_string[3] = 'r'
print(my_string)
```

```
hello
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-3-7587cc2651c9> in <module>
      1 my_string = "hello"
      2 print(my_string)
----> 3 my_string[3] = 'r'
      4 print(my_string)

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

We Get the above error because of the Immutability of the string Datatype

---

Home work

1) Take values of length and breadth of a rectangle from user and check if it is square or not.

2) Write a program to check for leap year. A leap year is exactly divisible by 4. Except for century year (i.e. divisible by 100), for a century year to be a leap year it should also be divisible by 400.

example inputs:

- 1000 is not a leap year
- 2000 is a leap year

3) Write a program for python calculator whose output depends on the users input operator and numbers. The program should prompt users to enter 2 numbers and also ask for an operator. The program should output the answer of the operation preformed.

Example:
Input 1 : 200
Input 2 : 400
Input 3 : +

OUTPUT : 600

4) Write the code to Split all the words and save them into a list

x = "Hello,How,Are,You,"

HOMEWORK SOLUTION

In [16]:

```
#TASK 1 :

length = int(input("enter length value: "))
breadth =int(input("enter breadth value: "))

if length ==breadth:
    print("It is square")

else:
    print("It is rectangle")
```

```
enter length value: 52
enter breadth value: 52
It is square
```

In [5]:

```
#TASK 2:

year = int(input("Enter a year: "))

if (year % 4) == 0:
   if (year % 100) == 0:
       if (year % 400) == 0:
           print("Leap year")
       else:
           print("Not a leap year")
   else:
       print("leap year")
else:
   print("Not a leap year")
```

```
Enter a year: 2000
Leap year
```

In [9]:

```
#TASK 3:
number1 = int(input("Enter first number: "))
```

```
number2 = int(input("Enter first number: "))
print("operations can be perfomred: +, -")
choice = input("Enter your choice:  ")

if choice =="+":
    print(number1+number2)


if choice =="-":
    print(number1-number2)
```

```
Enter first number: 56
Enter first number: 23
operations can be perfomred: +, -
Enter your choice:  +
79
```

In [10]:
```
#TASK 4:
x = "Hello,How,Are,You,"
y=x.split()
print(y)
```

In [ ]:

In [ ]: