

# Data Science with Python: Beginner to Advance



**Sheikh Fahim Faysal Sowrav**  
Researcher and Trainer, NOAMI

**MODULE-06**

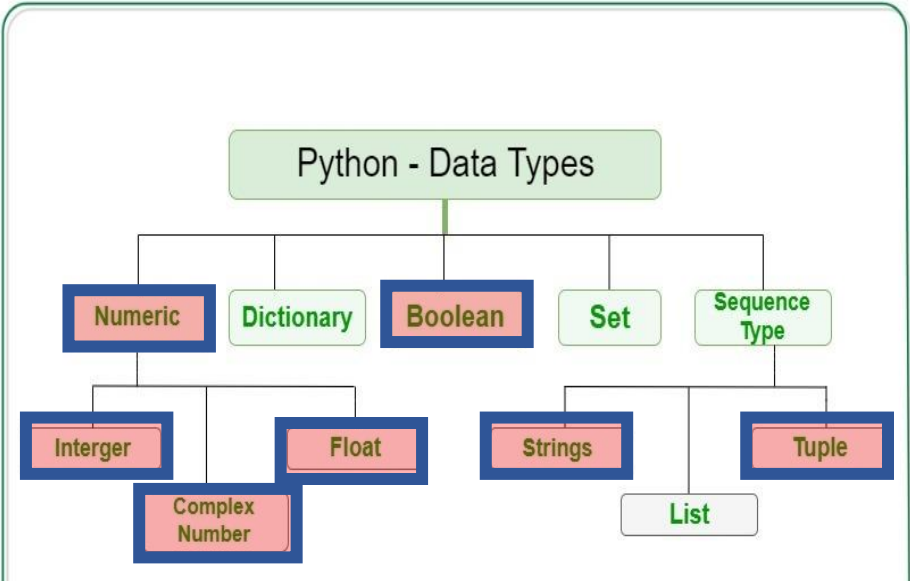


# RECAP

Operator	Meaning	Example	Result
+	Addition	5+3	8
-	Subtraction	5-3	2
*	Multiplication	5*3	15
/	Division	10/2	5
>	Greater Than	10 > 2	True
<	Less Than	2 < 10	True
==	Equal to	5 == 5, "Fahim" == "Fahim"	True
!=	Not Equal to	5 != 6, "Fahim" != "Faysal"	True



- Version and Software
- Installation
- Data Types
- Printing Text
- Simple Operations
- Home Task 01



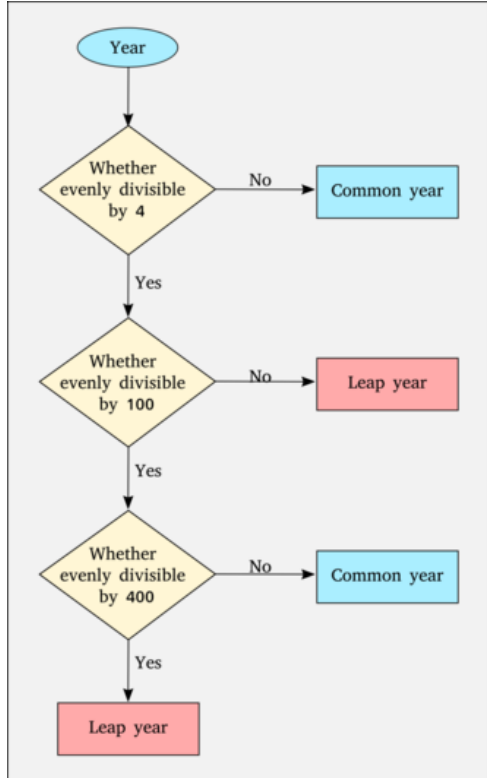
Name	Type	Description
Integers	int	Whole numbers, such as: 3 300 200
Floating point	float	Numbers with a decimal point: 2.3 4.6 100.0
Strings	str	Ordered sequence of characters: "hello" 'Sammy' "2000" "楽しい"
Lists	list	Ordered sequence of objects: [10,"hello",200.3]
Dictionaries	dict	Unordered Key:Value pairs: {"mykey" : "value" , "name" : "Frankie"}
Tuples	tup	Ordered immutable sequence of objects: (10,"hello",200.3)
Sets	set	Unordered collection of unique objects: {"a","b"}
Booleans	bool	Logical value indicating True or False

# RECAP

if Statements  
Nested if Statements  
else Statements  
Nested else Statements  
elif Statements  
Boolean Logics  
LISTS  
NESTED LISTS  
INDEXING like LIST in Strings

FLOATS  
EXPONENTIATION  
QUOTIENTS  
REMAINDER  
STRINGS  
BACKSLASHES  
NEW LINES  
CONCATENATIONS  
STRING OPERATIONS  
VARIABLES

  
**GitHub**  
 **replit**  
**INPUT**  
**Working with Input**  
**In-Place Operator**  
**Walrus Operator**  
**Booleans**  
**Comparisons**






# HOME TASKS 04

Solve all the Questions and Submit each of them with ONE python file: **(Code your own with you have learned from the course till now)**

- a) You have deposited 20,000 BDT in bank for a compound interest of 5% per year. Which means, after one year your balance will be your principal + profit. For the next year (principal + profit) will be counted as your new principal and profit will be calculated on your new principal. And this will go on. What will be your money after 4 years. **[Don't use the formula  $C=P(1+r/100)^n$ ] [Use in-Place operator]**
- b) Take all the following inputs of a user: **Name, Birth year, Nationality, University/College Name, Living Country, Male/Female, and Mobile number (11 digit)**. **DO NOT USE ANY IF ELSE or Advance CODE**

Then, give a output of his/her profile like following output 

**Name: Inputted Name here**

**Age: \*\*\* years**

**Nationality: \*\*\***

**University/College: \*\*\*\***

**Current Location: Inputted living country name**

**Mobile Number: +8801\*\*\*\*\***

**Gender: True (if male), False (if female)**

a

```

C = 20000
# After 1st year:
C += C * 0.05
# After 2nd year:
C += C * 0.05
# After 3rd year:
C += C * 0.05
# After 4th year:
C += C * 0.05
print('After 4 years the amount will
be: ',C)

```

b

```

name= input('Enter Name: ')
birthyear = int(input('Enter birth year: '))
nationality = input('Enter Nationality: ')
university = input('Enter University/College: ')
curLocation = input('Enter Current Location: ')
mobileNumber = input('Enter Mobile Number: +88')
gender = input('Gender: male or female:').lower()

print('-----')

print('Name: '+ name)
print('Age:', 2022-birthyear, 'years old')
print('Nationality: '+ nationality.upper())
print('University: '+ university)
print('Current Location: '+ curLocation)
print('Mobile Number: +88'+ mobileNumber[0:11])
print('Gender: ', 'male' == gender)

```

# HOME TASK 05

1. Make a code that will take input of two strings. If the two inputs are same, your program will print "SAME". If Not, It should show "NOT Same" and check which one is greater and print that.
2. Make a calculator, which will work as the following algorithm:
  - a. Input 1<sup>st</sup> Number
  - b. Input What you want to do with numbers (+, -, \*, or /)
  - c. Input 2<sup>nd</sup> Number
  - d. Do calculation
  - e. Show Result
3. Rainfall = [22, 3.4, 1, 8, 19, 21]  
From the above rainfall data, Find average rainfall of that area.
4. A school has following rules for grading system:
  - a. Below 25 - F
  - b. 25 to 45 - E
  - c. 45 to 50 - D
  - d. 50 to 60 - C
  - e. 60 to 80 - B
  - f. Above 80 - A

Ask user to enter marks and print the corresponding grade.



# Control Structure

# List Operations

The item at a certain index in a list can be reassigned. You can replace the item with an item of a different type.

```
nums = [7, 7, 7, 7, 7]
nums[2] = 5
print(nums)
```

Lists can be added and multiplied in the same way as strings. Lists and strings are similar in many ways - strings can be thought of as lists of characters that can't be changed.

```
nums = [1, 2, 3]
print(nums + [4, 5, 6])
print(nums * 3)
```

To check if an item is in a list, the `in` operator can be used. It returns `True` if the item occurs one or more times in the list, and `False` if it doesn't.

```
words = ["spam", "egg", "spam", "sausage"]
print("spam" in words)
```



# List Operations

To check if an item is not in a list, you can use the not operator in one of the following ways:

```
nums = [1, 2, 3]
print(not 4 in nums)
print(4 not in nums)
print(not 3 in nums)
print(3 not in nums)
```

# List Functions

The append method adds an item to the end of an existing list. For example:

```
nums = [1, 2, 3]
nums.append(4)
print(nums)
```

To get the number of items in a list, you can use the len function.

```
nums = [1, 3, 5, 2, 4]
print(len(nums))
```

Unlike the index of the items, len does not start with 0. So, the list above contains 5 items, meaning len will return 5.

# List Functions

The insert method is similar to append, except that it allows you to insert a new item at any position in the list, as opposed to just at the end.

```
words = ["Python", "fun"]
index = 1
words.insert(index, "is")
print(words)
```

The index method finds the first occurrence of a list item and returns its index. If the item isn't in the list, it raises a ValueError.

```
letters = ['p', 'q', 'r',
          's', 'p', 'u']
print(letters.index('r'))
print(letters.index('p'))
print(letters.index('z'))
```

There are a few more useful functions and methods for lists.

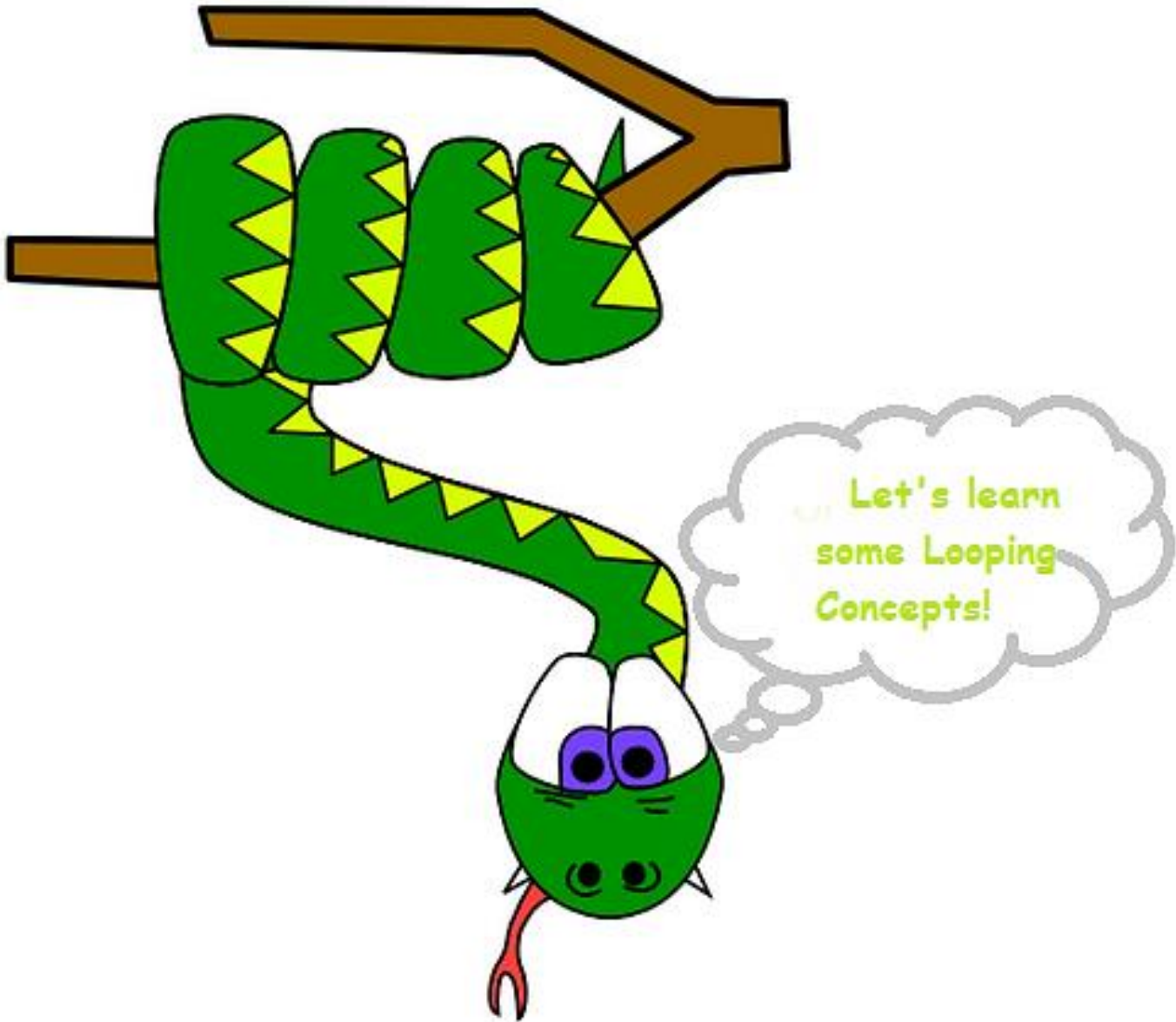
**max(list):** Returns the list item with the maximum value

**min(list):** Returns the list item with the minimum value

**list.count(item):** Returns a count of how many times an item occurs in a list

**list.remove(item):** Removes an object from a list

**list.reverse():** Reverses items in a list.



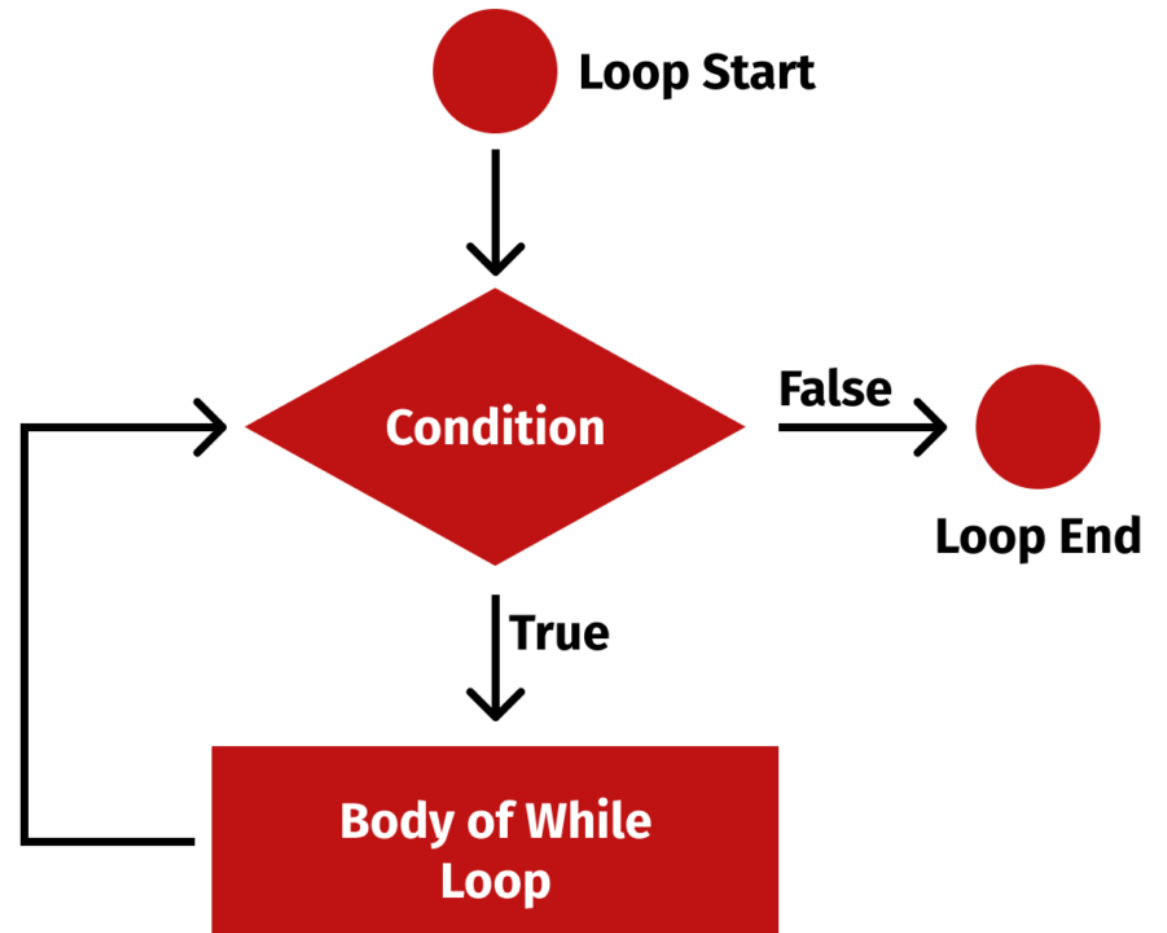
# while Loop

A while loop is used to repeat a block of code multiple times. For example, let's say we need to process multiple user inputs so that each time the user inputs something, the same block of code needs to execute.

Below is a while loop containing a variable that counts up from 1 to 5, at which point the loop terminates.

```
i = 1
while i <=5:
    print(i)
    i = i + 1

print("Finished!")
```





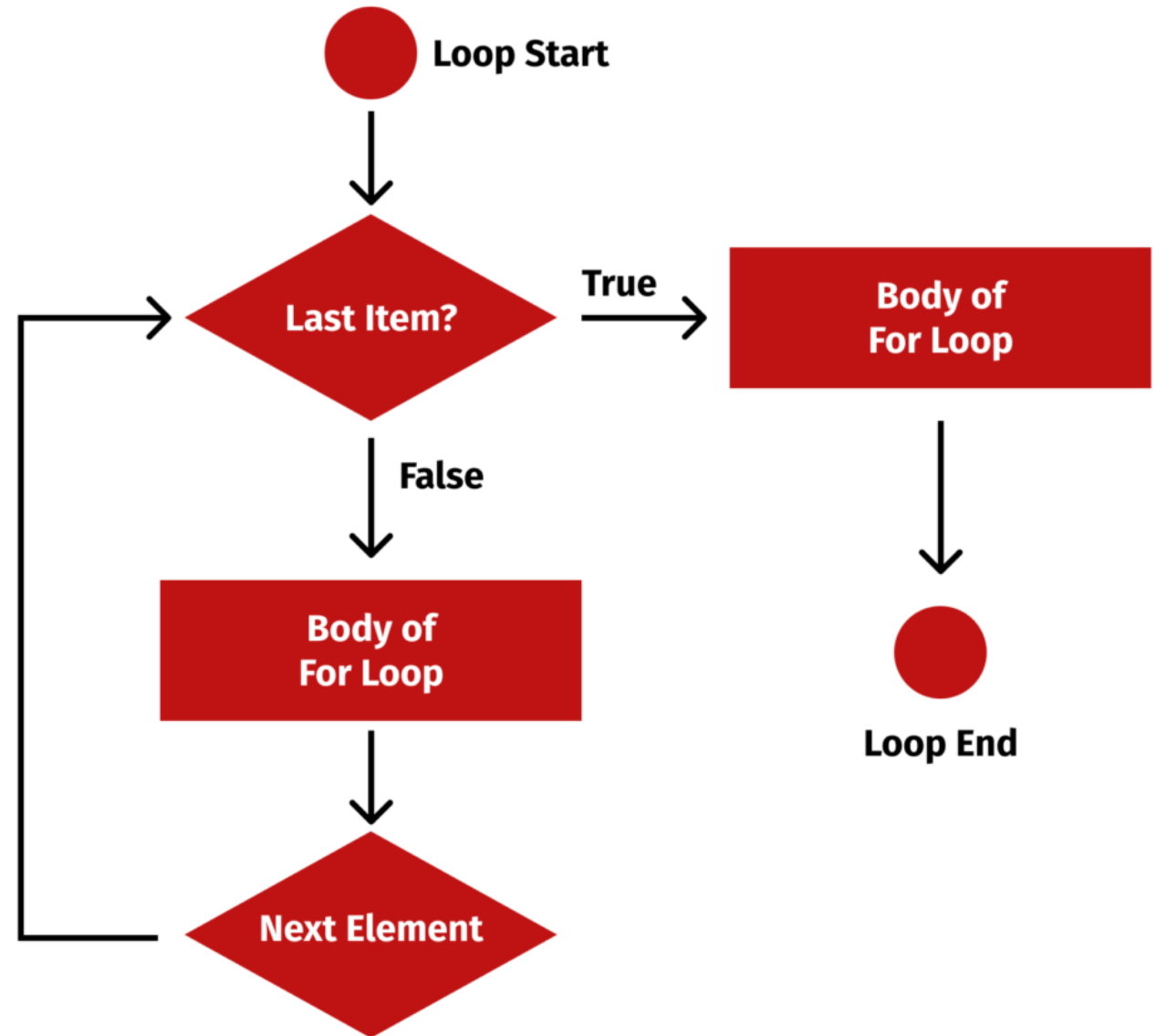
# for Loop

The for loop is used to iterate over a given sequence, such as lists or strings. The code below outputs each item in the list and adds an exclamation mark at the end:

```
words = ["hello", "world",  
"spam", "eggs"]  
for word in words:  
    print(word + "!")
```

The for loop can be used to iterate over strings.

```
str = "testing for loops"  
count = 0  
  
for x in str:  
    if(x == 't'):  
        count += 1  
  
print(count)
```



# for vs while Loop

Basis of Comparison	For Loop	While Loop
Declaration	for object in sequence: <indentation>body	while condition: <indentation>body
Format	Initialization, condition checking, and iteration statements are written at the top	Only initialization and condition checking is written at the top
Use Case	When you already know the number of iterations	When you don't know the number of iteration.
Iteration	Every element is fetched through the iterator/generator. Example, range()	Every element is explicitly incremented or decremented by the user
Generator Support	For loop can be iterated on generators in Python.	While loop cannot be iterated on Generators directly.
Disassembly	For loop with range() uses 3 operations. range() function is implemented in C, so, its faster.	While loop with incrementing variable uses 10 operations. i+=1 is interpreted, hence, it's slower than range()
Speed (May Vary on Conditions)	On basis of disassembly, for loop is faster than while loop.	On basis of disassembly, the while loop is slower than for loop.

# range

The `range()` function returns a sequence of numbers. By default, it starts from 0, increments by 1 and stops before the specified number.

The code below generates a list containing all of the integers, up to 10.

```
numbers = list(range(10))  
print(numbers)
```

If `range` is called with one argument, it produces an object with values from 0 to that argument. If it is called with two arguments, it produces values from the first to the second.

```
numbers = list(range(3, 8))  
print(numbers)  
  
print(range(20) == range(0,  
20))
```

`range` can have a third argument, which determines the interval of the sequence produced, also called the step. We can also create list of decreasing numbers, using a negative number as the third argument, for example

**`list(range(20, 5, -2)).`**

The for loop is commonly used to repeat some code a certain number of times. This is done by combining for loops with range objects.

You don't need to call `list` on the range object when it is used in a for loop, because it isn't being indexed, so a list isn't required.

```
for i in range(5):  
    print("hello!")
```

# break

To end a while loop prematurely, the break statement can be used. For example, we can break an infinite loop if some condition is met:

```
i = 0
while 1==1:
    print(i)
    i = i + 1
    if i >= 5:
        print("Breaking")
        break

print("Finished")
```

```
fruits = ["apple", "banana", "cherry"]

for x in fruits:

    print(x)
    if x == "banana":

        break
```

# continue

Another statement that can be used within loops is continue. Unlike break, continue jumps back to the top of the loop, rather than stopping it. Basically, the continue statement stops the current iteration and continues with the next one.

```
i = 0
while i<5:
    i += 1
    if i==3:
        print("Skipping 3")
        continue
    print(i)
```

```
fruits = ["apple", "banana", "cherry"]
for x in fruits:
    if x == "banana":
        continue
    print(x)
```



# CLASS WORK

- 1. Iterate through every number in a list to separate out even and odd numbers. Identify possible outlier values as well.**
- 2. You have a data of last 20 years average rainfall in Sylhet. You are trying to categorize them in three types, Heavy Rainfall, Medium Rainfall and Low Rainfall. Make a short code that can be used for any data we input, it make them in three categories and make three separate lists.**
- 3. You have the same data of last 20 years average rainfall in Sylhet. You are trying to categorize them in three types, Heavy Rainfall, Medium Rainfall and Low Rainfall. But now you want an additional list of only Leap years data in a list.**

# HOME TASK 06

1. **Make a Do or DIE game of your own ! For help, see the example at given Github link.**
2. **yearly\_temp = [[2016, 2017, 2018, 2019, 2020, 2021],[28, 30, 27.5, 30.5, 31, 29.9]]**  
**Find the average temperature of last six years (2015-2021), Find the maximum temperature and it's year.**

# RESEARCH PROBLEM

**Teams will be made on next class**