Lists

- A list is a sequential collection of Python data values, where each value is identified by an index.
- The while statement keep running indefinitely unless it comes across a return or break statement.
- The values in a list do not have to be of the same type.
- A list within another list is said to be nested and the inner list is often called a sublist. There is a special list that contains no elements. It is called the empty list and is denoted [].
- Negative index values will locate items from the right instead of from the left.
- In and not in are used to check for list membership eg fruit = ["apple", "orange", "banana", "cherry"]print("apple" in fruit)
- The + operator concatenates listseg.print([1, 2] + [3, 4])
- In Python, every object has a unique identification tag id alist = [4, 5, 6] id(alist)
- Unlike strings, lists are mutable. This means we can change an item in a list by accessing it directly as part of the assignment statement. Eg fruit = ["banana", "apple", "cherry"]fruit[0] = "pear"
- The del keyword is used when you want to delete an item in the list eg a = ['one', 'two', 'three'] del a[1]print(a)
- del handles negative indices and causes a runtime error if the index is out of range.
- Cloning a list eg 1 a = [81, 82, 83] b = a[:]
- The split method breaks a string into a list of words. By default, any number of whitespace characters is considered a word boundary.
- An optional argument called a **delimiter** can be used to specify which characters to use as word boundaries. The following example uses the string ai as the delimiter:
- The join is used to join together words, and by specifying a delimiter and the join removes the list if you were dealing with lists .for example
  wds = ["red", "blue", "green"]
  glue = ','
  s = glue.join(wds)
  print(s)
- Giving the same name to 2 lists with the same value is called aliasing and making changes in one affects the other eg a = [81, 82, 83] b = a print(a is b)
- To convert a word into a list just use the word list and every word is converted to a to a list of characters eg xs = list("Crunchy Frog")
- Append adds an item at the end of the list.

- Python tries to avoid duplicates values in memory to avoid that it creates object that can be referenced using variables.
- a list is a collection of references to objects eg a is a reference to object 81
- Adding two lists together both require be to be lists i.e concatenation.origlist = [45, 32, 88] origlist = origlist + ["cat"]
- This example gets the fruits by their index throughout the length of the fruits fruits = ["apple", "orange", "banana", "cherry" for position in range(len(fruits)): print(fruits[position])
- A **pure function** does not produce side effects. It communicates with the calling program only through parameters (which it does not modify) and a return value. Eg using a global variable changes the output hence not a pure function
- Functions which take lists as arguments and change them during execution are called modifiers and the changes they make are called side effects.

- Accumulator in lists to get the min and max value without updating.Reasigning the bestnum to a new value

```
nums = [9, 3, 8, 11, 5, 29, 2]
best_num = 0
for n in nums:
if n > best_num:
  best_num = n
print(best_num)
```

[<expression> for <item> in <sequence> if <condition>]
It is called a list comprehension in Python.

What it does:
It builds a new list by:

1.Looping through each <item> in <sequence>.

2. Optionally filtering the items with the if <condition> part (only items that meet the condition are used).

3.Applying an expression (<expression>) to each selected item, and putting the result into the new list.

- A nested list is a list that appears as an element of another listeg

```
nested = ["hello", 2.0, 5, [10, 20]]
innerlist = nested[3]
print(innerlist)
item = innerlist[1]
print(item)

print(nested[3][1])
```

In nested list if you want to check for an item in the nested list you would do this as an example.

```
alist = [ [4, [True, False], 6, 8], [888, 999] ]
```

alist[1] = [888, 999]
So alist[1][0] = 888.