while .. else and for..else

Syntax:	**Syntax:**
while <condition>: **statement-1** **statement-2** **else:** **statement-3** **statement-4**	**for variable in <iterable>:** **statement-1** **statement-2** **else:** **statement-3** **statement-4**
statement-1,statement-2 are repeated until given condition is True, if condition is False it executes else block	statement-1,statement-2 are repeated until read all the values from iterable. After reading the all the value, python executes else block

Note: else block is not executed if while loop or for loop is terminated using break statement.

Example:

```
n=1
while n<=5:
    print("Inside While")
    n=n+1
else:
    print("Inside Else Block")

for n in range(1,6): # 1 2 3 4 5
    print("Inside for block")
else:
    print("Inside Else Block")
```

Output

```
Inside While
Inside While
Inside While
Inside While
Inside While
Inside Else Block
Inside for block
Inside for block
Inside for block
Inside for block
Inside for block
Inside Else Block
```

Example:

```
n=1
while n<=5:
    print("Inside While")
    n=n+1
    break
else:
    print("Inside Else Block")

for n in range(1,6): # 1 2 3 4 5
    print("Inside for block")
    break
else:
    print("Inside Else Block")
```

Output

```
Inside While
```

Inside for block

Example:

```
# Factorial of number
num=int(input("Enter any number "))
fact=1

while num>0:
    fact=fact*num
    num=num-1
else:
    print(f'Factorial is {fact}')
```

Output

```
Enter any number 0
Factorial is 1

Enter any number 3
Factorial is 6
```

Collection Data types or Data Structures

Python data types are classified into 2 categories

1. Scalar Data Types
    a. Int
    b. Float
    c. Complex
    d. Bool
    e. NoneType

Scalar data types are used to represent one value in memory.

2. Collection Data Types

a. Sequences
    i. List
    ii. Tuple
    iii. String
    iv. Range
    v. Bytes
    vi. Bytearray
b. Sets
    i. Set
    ii. frozenset
c. Mapping
    i. Dict

Collection data types are used to represent more than one value. Collection data types are used to group individual objects into one object.

1. Avoiding creating of number of variables for holding number of value
2. The values exists within collection are referred with one name
3. By grouping all the values we can perform aggregate operations.
4. Transporting data from one place to another place
5. Collection uses data structure for storing data. Data structure define set of rules and regulations for organizing data in memory

Based on organization of data, these collections are classified into 3 categories

Sequences

In sequences or sequence data type's data is organized in memory in sequential order (one by one).

Advantage of sequences

1. Ordered Collections, where insertion order is preserved or same.
2. Read and Write sequential or randomly
3. Allows duplicate values
4. Homogenous or Heterogeneous
5. Sequences are index based collections

In Python, an ordered collection is a data structure where the elements are stored in a specific sequence, and their position matters. The order of elements is preserved, meaning the order in which they are added or inserted is maintained when you access or iterate over them.

Sequences are classified into 2 categories

1. Mutable Sequences
    a. List
    b. Bytearray
2. Immutable sequences
    a. Tuple
    b. String
    c. Bytes
    d. Range

List

List is ordered collection or mutable sequence data type
After creating list changes can be done.
List allows duplicate value

List allows reading in sequential or random
List is index based collection where reading and writing is done using index.
List can be Homogenous or Heterogeneous
In application development list is used to group individual objects or values, where duplicates allowed and reading and writing is done sequentially and randomly.

How to create list?

List can be created in different ways

1. Empty list can be created using empty square brackets []
2. List can be created with values separated with comma with in square brackets [value,value,value,…]
3. List can be created using list comprehension
4. List can be created using list type or function
    a. List()
    b. List(iterable)

Note: Every collection type is called iterable. Iterables allows to read individual elements or items.

```
>>> A=[]
>>> print(A,type(A))
[] <class 'list'>
>>> B=[10,20,30,40,50]
>>> print(B,type(B))
[10, 20, 30, 40, 50] <class 'list'>
>>> C=[1.5,2.5,3.5,4.5]
>>> print(C,type(C))
[1.5, 2.5, 3.5, 4.5] <class 'list'>
>>> D=["naresh","suresh","ramesh"]
>>> print(D,type(D))
```

['naresh', 'suresh', 'ramesh'] <class 'list'>
>>> E=[101,"naresh",5000.0,True]

Example:
Write a program to represent student marks details

```
rollno=1
name="naresh"
marks=[60,50,70]
print(rollno,name,marks)
print(type(rollno),type(name),type(marks))
```

Output
1 naresh [60, 50, 70]
<class 'int'> <class 'str'> <class 'list'>

List can be created using list() function, this function to convert other collection types into list.

Syntax1: list() Create empty list
Syntax2: list(iterable) This create list using values from existing collection or iterable

Example:
```
>>> A=[]
>>> B=list()
>>> print(A,B)
[] []
>>> C=list(range(1,6))
>>> print(C)
[1, 2, 3, 4, 5]
>>> D=list(range(10,60,10))
```

```
>>> print(D)
[10, 20, 30, 40, 50]
>>> E=list("NIT")
>>> print(E)
['N', 'I', 'T']
>>> F=list([10,20,30,40,50])
>>> print(F)
[10, 20, 30, 40, 50]
```

How to read data or elements or values from list?

Any sequence data can read in different ways

1. Indexing
2. Slicing
3. for loop
4. iterator
5. enumerate

indexing

What is index?

An index integer value

Each value or item in sequence/list is identified with unique number called index.

This index can be +ve or –ve

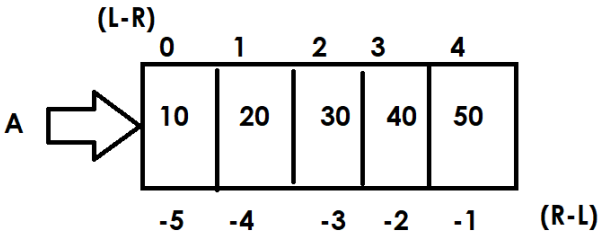
+ve index start at 0, which is used to read data from left to right

-ve index start at -1, which is used to read data from right to left

This index is used as subscript to read and write data in list

Syntax: list-name[index]

A=[10,20,30,40,50]



A[0] --> 10
A[1] --> 20
A[2] --> 30
A[3] --> 40
A[4] --> 50
A[5] --> IndexError

A[-1] --> 50
A[-2] --> 40
A[-3] --> 30
A[-4] --> 20
A[-5] --> 10
A[-6] --> IndexError