Lists:-

used to store data under a single name with any data types. It can store as many values we need.

eg:-

student = ["karthik", "Rahul", "sai"].

print(student) → displays the entire content.

print(student[0]) => karthik
print(student[1]) = Rahul
print(student[2]) = sai

print(student[-1]) = sai
print(student[-2]) = Rahul
print(student[-3]) = karthik.

Print(student[1:]) → Prints the entire elements from index 1
print(student[0:2]) → prints elements in that range.

List functions:-

(a) extend function :- It would append the elements to the list

eg:- student = ["karthik", "Rahul", "sai"]
id = [27, 43, 2]
student.extend(id)
print(student)
o/p = ['karthik', 'Rahul', "sai", 27, 43, 2].

→ or we can append the data required

student.extend("Balu")
print(student)
o/p = ["karthik", "Rahul", "sai", "Balu"]

If we want to add elements into middle of the list we were having a fn known as insert.

Eg:- Listname.insert (indexno, new element)

student.insert (0, Bobby)

print (student) → O/P = ["Bobby", "karthik", "Rahul", "hi", ...]

we can remove the element from list by giving the name of element.

student.remove("karthik")

Print (student) → O/P = ["Bobby", "Rahul", "sai", "Balu"]

we can clear entire list by clear method.

student.clear() → removes every elements and returns empty block ([])

↑
O/P

* Pop function: removes the last element in list

student.pop() → O/P => Balu

* finding element in index.

print (student.index("sai") → O/P = 2

Syntax:- Listname.index ("element to be searched").

If element is not present it return error.

* counting the similar elements in list.

Syntax:- listname.count (element)

eg:- print(~~student("Bo~~ student.count ("Balu")) → O/P is 1

since we were having one element

Sort :-

It is used to sort the list

Syntax:- listname.sort()

print(id.sort())

↳ o/p = [2, 27, 43]

Reverse list :-

It is used to reverse the given list

syntax- listname.reverse()

print(id.reverse()) → o/p : [2, 43, 27]

copy list :- copies the list

destination list = source list.copy()

friends = student.copy()

print(friends) → o/p = ["karthik", "Rahul", "Balu"]

Tuples (brackets are diff)

↳ It is similar to lists and acts as a container to store data, only some key features may change.

↳ It can't be changed or modified, simple it is declaration.

↳ we can access elements but cannot change or modified.

eg:- student = ("karthik", "Rahul", "sai")

Print(student[0]) → o/p = karthik (we can just access data but cant modify them).

functions:-

↳ def is the keyword for the functions

def fnname ( ) : → represents that all the code comes after this line is inside the fn

fn defination

~~fn call~~ functioncall ()

passing through parameters

def fnname (parameter 1, parameter 2 - - - -) :

fn defination

fn call (parameter - -)

Return statements:-

def fnname (parameter) :

return data

print (fnname or fn call)

conditional parmeters. (if or ifelse)

↳ for if we need a boolean variable

↳ IF boolean_variable :

defination of condition that is needed to execute

else

another condition needed to be executed.

elseif is written as elif

logical or = or

logical and = and

logical not = not

↳ negeate what ever in it

checking and comparision:-

if condition: (eg: num1 ≥ num2)

elif condition

statements conditions | for any types i.e strings, Numbers and values we can go through this method. (==)
↳ return true if both of them are equal

Dictionaries in python:-

↳ special structure in python that allows us to store key value pairs.

↳ we can access information through the key

↳ give name to dictionaries, we should not need to duplicate dictionary we can use any of the keywords.

eg:- monthconversions = {
↓
dictionary name

data → key
"Jan": "January",
↳ data.

:

}

function call :-

eg: print (monthconversions [keyname])

or

monthconversions.get("keyname")

If we donot have key that is matched it returns none

or. monthconversions.get ("keyname", to be printed if it is invalid.

Loops:-

While_loop:-

(indented only would be considered into the loop

While condition :

| statements

for_loops:-

In case of strings

for letter in " ": ↳ tells that for each letter iteration takes place.

```
friends = ["Jim", "karen", "kevin"]
for anyname in friends:
    print(anyname)
```

↳ for length we would write

```
len(friends)):
    print (friends[index])
```

```
for x in range(10):
    print(x) #prints every number from 0 to 9.
```

↳ can be written as range (starting, en

exponent function:-

eg: print (2**3) #prints in such a way that 2 to power 3

2D-lists

numbers here we would add lists in a multiple ways

```
eg:- lists = [
    [a, b, c],
    [d, e, f],
    [g, h, i],
    [j]
]
```

SHEET No.

accessing these elements are

```
print (listname[ ][ ])
```

→ index of column (second brackets)
↳ Index of row (first brackets)

```
for anyvariable in listname:
    print(x) → prints the entire elements in list
```

```
for row in listname:
    for colours in row:
        print (col)
```

Basic translator:
we would work through converting the ove' vowels into the character 'g'.

```
IF letter in " string":
```

↳ this would say that letter is present in string or not

<u>Try/catch statements:-</u>
we know that try/catch statements would help us to handle various types of errors.

```
try:
    statements
except:
    print statements
```

// by zero exception / except zeroDivisionError as err:
↓ zeroDivisionError: error type
↓ err: variable used to store the particular error

// ValueError used to store input format errors

Reading and writing files:-

Reading

var = open("filename", "r")

r - read mode
w - write mode
a - append

When we open a file we should need to close.

var.close()

we can print the entire file

initially we need to know whether the file is readable or not for this we use var.readable()

↓ tells whether the file is readable is not

we can write and read only file.

for reading variable.nable.readline()

prints the first line

we can use variable_name.readlines()

↓ gives the entire file

variablename.readline()[ ]

↓ Index of the line

by for loop :-

for ~~emplo~~ variablename in variablename.readlines()

print(variable name)

print(~~read~~ variablename.read()) → prints entire content of file.

Random fn
↳ random.randint (start, stop)

SHEET No.

PRAGNANAM BRAHMA
ANITS

→ we would apply mode through writting and appending

above given r, w, a modes.

variablename.write ("string type of file")

Modules anp pip:- ↳ it is used to install python modules

↳ a python file which we can import to our current python file so that we use those all in our current prg

it is similar to packages in java.

syntax:- import file_name

~~print~~ we can call those functions through file-name.fncall

classes and objects :-

class ___ :
name of class

// statements, we can declare types.

we need to initialize fn

def __init__(self) : ← we can have initialize fn and can mapout what a student can have.

(self, , , , ,)

we can have parameters in it

self.parameter = parameter

Accessing the classes :- we need to import that class & creat object

from student import student

filename classname

object = student( )

with parameters

similar to java we can access the ~~et~~ context by with the help of the objectname.parameters

the parameters we were passing are moved to the init function and assigning them.