- **Assertion Errors**: The assert keyword lets you test if a condition in your code returns true, otherwise it raise assertion error. When the condition is false.

eg: x = 'Hello'
assert x == 'Hello'

Unit-2
Function Basics

Syntax to define a function:

Keyword def function_name (Parameters):
    # Statement
    # Statement
    return expression

(calling) function_name (arguments)

eg.
```
def even_odd():
    if a%2==0:
        print("even")
    else:
        print("odd")
a=int(input("enter a"))
even_odd()
```

Output:
enter a 5
odd

Date / /
Page:

⇨ Types of arguments in functions:

1. Positional arguments
2. Keyword argument
3. Default arguments
4. Arbitrary argument
   - *args
   - **kwargs

1. **Positional arguments:** We use these arguments during the function call, so that the first argument is assigned to first parameter and the second argument is assigned to second parameter.

eg:
```
def info(name, age):
    print("My name is", name)
    print("My age is", age)
info("xyz", 18)
```
→ order matters

Output
```
My name is xyz
My age is 18
```

2. **Keyword arguments:** The idea is to allow the ~~color~~ caller to specify the argument name with values so that the caller does not need to remember the order of parameters.

eg
```
def info(name, age):
    print("My name is", name)
    print("My age is", age)
info(name="xyz", age=18)
```
→ order doesn't matter

Output
```
My name is xyz
My age is 18
```

Teacher Signature ..... Parents Signature .....

Date / /
Page

* Mixture of Keyword & Positional Argument.

eg
```
def info(name, age):
    print('My name is', name)
    print('My age is', age)
info('xyz', age=18)
```

Note: Keyword argument always comes after the positional argument.
If we write the positional argument before the keyword argument then it will give us error.

info(Name='xyz', 18) ——— X

3. Default argument: A default argument is a parameter that assumes a default value if a value is not provided in a function call for that argument.

eg: ~~eg def add~~
```
def add(x, y=50):
    print('The value of x', x)
    print('The value of y', y)
    print(x+y)
```

(i)
```
add(10)
```
Output:
The value of x 10
The value of y 50
60

(ii)
```
add(10,20)
```
The value of x 10
The value of y 20
30

Teacher Signature .....

Parents Signature .....

Date / /
Page:

eg2. def add (x, y, z=50):
print ("The value of x", x)
print ("The value of y", y)
print ("The value of z", z)
print (x+y+z)
add (10, 20)

Output The value of x 10
The value of y 20
The value of z 50
80

4. Arbitrary argument: ~~Variable length argument~~
(i) *args → Non Keyword argument
(ii) **kwargs → Keyword argument

eg def add (*numbers): → collection of elements stored in Tuple
c = 0
for i in numbers:
c = c+1
print (c)
add (4, 7, 3, 5, 10)

Output 29

Teacher Signature .....
Parents Signature .....

→ collection of elements stored in dict.

eg: def info (**kwargs):
    for key, value in kwargs.items():
        print(key, value)
info (First = 'xyz', Second = 'hlo')

* Return Statement: It is used inside a fun to send the functions result back to the caller (fun call).

def add (a, b):
    c = a+b
    return c
    print ("Hello") → The code after return statement will not be executed
print (add (6,5))

Output 11

* Return multiple values:

eg def add():
    a = 'xyz'
    b = 'hlo'
    return a, b
print (add())

output ('xyz', 'hlo')

* Nested function: def f1():
    s = 'hlo'
    def f2():
        print(s)
    f2()
f1()

Output hlo

Teacher Signature ..... Parents Signature .....

Date / /
Page:

Types of functions :

1. Built-In function
   - Print ()
   - len ()
   - max ()
   - factorial ()
   - min ()
   - Pow ()
2. User defined function
   - def add (x, y)

- **Call by Value** : When immutable objects such as integers, strings are passed as arguments to the function call then it can be considered as call by value.

eg1
```
def change (a):
    a = a + 10
    print (a)
a = 10
print (a)
change (a)
print (a)
```

Output : 10
20
10

Note : Change inside the function can not reflect outside the function

Teacher Signature .....

Parents Signature .....

Date / /
Page

eg 2
```
def change (s1):
    s1 = 'xyz'
    print (s1)
s1 = 'abc'
print (s1)
change (s1)
print (s1)
```
Output : xyz
abc

- **Call by Reference :** These are mutable such as list etc.

eg
```
def change (list):
    list [0] = list [0] + 10
    print (list)
list = [10]
print (list)
change (list)
print (list)
```
Output 10
20
20

Note : Any change inside the function reflect inside the function.

⇨ **Math Module in Python :**

Constants in math Module.

Syntax to import math module :

```
import math
```

Teacher Signature .....

Parents Signature .....

Date / /
Page:

(i) Euler's Number

$e = \sum_{n=0}^{\infty} \frac{1}{n!}$ print (math.e)

$= \frac{1}{1!} + \frac{1}{2!} + \frac{1}{3!} ---- \frac{1}{n!} = 2.71$

(ii) Pi → print (math.pi) o/p = 3.14 ---

(iii) Tau → 2π print (math.tau) · o/p = 6.2831 ---

(iv) Infinity → print (math.inf) o/p = inf
print (-math.inf) o/p = -inf

(v) NaN (Not a number) → print (math.nan)
o/p → nan

Mathematical functions in Math Module :

- dir (math) → function that are present in math module.

ceil()

- x = 5.6 #6 gives bigger value.
  ↳ print (math.ceil (x))
- x = 5.6 #5
  ↳ print (math.floor (x)) gives small value

copysign

- x = 5
  y = -2 #-5.0
  print (math.copysign (x, y)) copy the sign of y

Teacher Signature ..... Parents Signature .....

Date / /
Page:

5. x = -5 # 5.0 gives the +ve value of -ve value.
print (math.fabs(x))
absolute

• x = 5 #120 gives the factorial
print (math.factorial(x))

• x = 5 #1.0
y = 2
print (math.fmod(x, y))
modulus

Pow() • x = 5 #25.0 gives the exponent value.
y = 2
print (math.pow(x, y))

Sqrt() • x = 25 # 5.0 gives the square root
print (math.sqrt(x))

⇨ Random Number Generator in Python
There are total 8 methods to generate the random numbers in Python.
import random

1. random() → range (0, 1)
n = random.random() # 0.68
print (n)

2. randint() [always give int value]
n = random.randint(40, 100) # 80
print(n)
min max

Teacher Signature ..... Parents Signature .....

Date / /
Page.

How to print multiple numbers with randint.

eg
```
import random
rand_list = [ ]
for i in range (0,5):
    n = random.randint (40,100)
    rand_list.append (n)
print (rand_list)
```
61, 80, 48, 58, 90

3 Sample ( ) : It will generate a sample of random integers within a given range.

eg
```
import random
n = random.Sample (range (40,100), 5)
print (n)
```
41, 58, 69, 70, 80

4. Choice ( ): Used for String, Tuple and List.

```
import random
lst = [1,5,7,8,9]
n = random.choice (lst)
print (n)
```
1

Teacher Signature .....

Parents Signature .....

Date / /
Page

5. randrange()

eg import random
n = random.randrange(50, 100, 4)
print(n)

start, stop, step

[50, 54, 58 ... 100]

58

6. Uniform(): It gives the floating value.

eg import random
n = random.uniform(15, 20)
print(n)

16.39

7. Shuffle()

eg import random
lst = [1, 2, 3, 4, 5]
a = random.shuffle(lst)
print(lst)

[2, 4, 1, 5, 3]

8 Seed() → initilize the random no. generator

eg import random
val = int(input("enter val"))
random.seed(val)
for i in range(val):
    print(random.random())

enter a 2
0.896
0.8834

Teacher Signature .....

Parents Signature .....

Date / /
Page:

⇨ Import Keyword in Python

Import is a keyword used to import different modules.

eg `import datetime`
(import: keyword, datetime: module)

```
print (datetime.datetime.now())
```
(datetime: module name; datetime.now(): method in module)

Output 2024-03-06 09:03:50.875589

⇨ Namespaces and scope

Namespace is a collection names like function name, variables

Types: 1. Local Name space
2. Built in Name space eg print(), id()
3. Global Name Space

eg
```
global_var=10
def outer_fxn():
    outer_var=20                    # 10
    def inner_fxn():                # 20
        inner_var=30                # 30
        print(inner_var)
    print(outer_var)
    inner_fxn()
print(global_var)
outer_fxn()
```

Teacher Signature .....

Parents Signature .....

Date / /
Page

Scope: It refers to coding region from which a particular python variable is accessable

eg
```
def greet():
    message = 'Helo'  → local variable
    print(message)
greet()                    # Helo
```

eg
```
Message2 = 'bye'
def greet():
    message = 'Hlo'        # Hlo
    print(message)         Bye
    print(Message2)        Bye
greet()
print(Message2)
```

⇒ Exception Handling

Exception is an incident that happens while executing a program that causes the normal flow of program to be interupted. An object in python that describes an error is called exception

eg
```
a = 5                  # 2.5
b = 2                  Bye
print(a/b)
print("bye")
```
[Normal flow of execution without error]

Teacher Signature .....

Parents Signature .....

Date / /
Page:

⇨ Type of Keywords in Exception Handling

1. Try: The try block lets you test a block of code for errors.

2. Except: It lets you to handle the error.

3. Else: The else block lets you to execute the code when there is no error in try block

4. Finally: It lets you execute the code regardless of the result of try and except block.

eg:
```
a=5
b=0                          # There is an error
try:                             division by 0
    print(a/b)                   bye
except Exception as e: → This tell the name of error
    print('There is an error')
    print(e)
print('bye')
```

Note: Except case only execute when try case is false or there is error in try.

Teacher Signature ..... Parents Signature .....

Date / /
Page

eg:
```
a = 5
b = 0
try:
    print ('Resource Open')
    print (a/b)
    print ('Resource closed')
except Exception as e:
    print ('wrong')
    print (e)
finally:
    print ("Resource closed 2")
```

Resource Open
wrong
division by zero
Resource Closed 2.

eg:
```
a=5
b=2
try:
    print ('Resource open')
    print (a/b)
    print ('Resource closed')
except Exception as e:
    print ('wrong')
    print (e)
else:
    print ('exit')
```

Resource Open
2.5
Resource closed
exit.

Note: Else block will only execute when there is no error in try block.

Teacher Signature .....

Parents Signature .....