Python Functions :- A function is a block of code which only runs when it is called. You can pass data, known as parameters, into a function. A function can return data as a result.

→ Creating a function :

• To call a function, use the function name followed parenthesis:

```
Keyword
def my_function():
    print("Hello ")

my_function()  # Calling a function
```

→ Arguments: Information can be passed into functions as arguments.

→ Arguments are specified after the function name, inside the parenthesis. You can add as many as you want, just separate them with a comma.

```
def my_function(fname): # (arguments)
    print(fname, 'Refsnes')

my_function("John")
```

The terms parameter & arguments can be used for the same thing: Infⁿ that are passed in Function.

Day-7

Date : 11 / 07 / 24

MON TUE WED THR [✓ FRI] SAT SUN

Subject Python

Return : To let a function return a value, and end the execution of the function call.

```
def my_function(x):
    return 5* x

print(my_function(3))   # 15
```

Pass : Functions defination cannot be empty, but if you for some reason have a function definition with no content, put in the pass statement to avoid getting an error.

```
def my_function():
    pass
```

PYTHON MODULES :- Consider a module to be the same as a code library.

→ A file containing a set of functions you want to include in your application.

→ To create a module just save the code you want in a file with the file extension .py :

ex. Save this code in a file named mymodule.py

```
def greeting(name):
    print("Hello", name)
```

→ Now we can use the module we just created, by using the import statement:

```
import mymodule
mymodule.greeting("Jonathan")
```

⇒ Renaming a Module:
You can create an alias when you import a module, by using the as keyword:

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```
} Renaming a Module

⇒ Security in making Python Module:-

→ We can use the "if" condition in the file to make the data secure:
→ We can use:

```
if __name__ == "__main__":
    # code / data
```
} The infⁿ inside this condⁿ will not be accessed by another file.

→ The infⁿ inside this condition will only be executed directly when this file is compiled.