FUNCTIONS:

Topic

- Function
- Patterns

Functions

- ➔ It’s a construct available in program language by which we can reduce duplicity in code and make our code easy to understand.
- ➔ Code wrapped in small bodies which we would be invoking/calling.

Ex:

print (“hello”)
print (“welcome to class”)

If we are to call this function again, we use loop, or copy it. So, using function,

```
example 1
def greet ():
    print ("Hello")
    print ("Welcome to this class ")
# Greeting
# greet () # calling a function
print ("press something ")
print ("abcdd")
greet ()
print ("adfsdf")
print ("fdsfsdf")
greet ()
```

OUTPUT:

```
press something
abcdd
Hello
```

```
Welcome to this class
adfsdf
fdsfsdf
Hello
Welcome to this class
```

Once a function is created, it can be called using the <name_of_the_function>.

Instead of repeating the print, we can use the function to call it.

Ex:

```python
def cubetilln ():

    n = 5

    sum = 0

    for i in range (1, n + 1):

        sum += i ** 3

    print (sum)

cubetilln () #call
```

OUTPUT

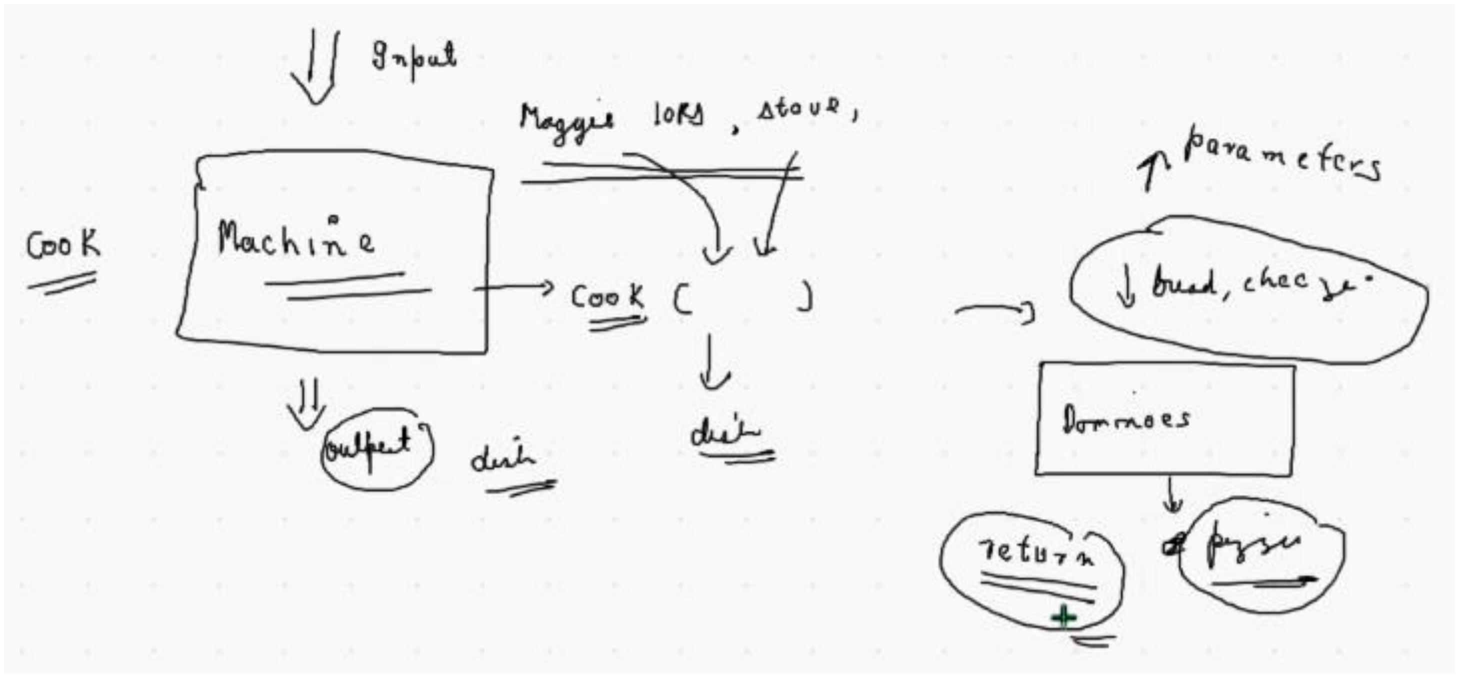
225

People can read the name and understand what it is for.

```python
def greet ():

    print ("Please enter some no: ", end="")

    N = int (input ())

    print ("Good Morning!!")

    for x in range (0, N):

        print (x)

def walk ():

    print ("Walking!!")

def sleep ():

    print ("Sleeping")
```

```
walk ()

sleep ()

greet ()

sleep ()

walk ()
```

Functions are like a machine. For example, in a food processing unit, if we provide some materials, it will give us food as output.



Input →   Bread, cheese, →*parameters*

Dominos

Output →   Pizza   → *return*

A function can take inputs, which are called parameters to the function. The parameters are the types pf python.
Eg: int, float, str, list, etc…

A function can return some data, which also are of types.
Eg: int, float, str, list, etc…

<u>Ex:</u>

This function will take two numbers as input and will give result output as sum. The input is passed inside this def as no1 and no2. If you want this function to return the sum of no1 and no2, you say “*return no1 + no2*”

To supply parameters (input) to the function we add the variables which we want as input inside the parenthesis. To get the result, we use the keyword *return* then the thing which we want to return. For example, in the below case, sum of no1 and no2:

```
def addtwonumbers (no1, no2):     → input
    return no1 + no2              → output
```

def addToNo(a, b)
	return a+b

in this when we give (5, 10), the a will be mapped to 5 and b will be mapped to 10. Then the result will be 5 + 10 = 15.

addToNo (5, 10)

- Print – prints the value
- Return – throws some value which needs to captured using a variable.

There is not need to put an argument in the function, but if you put an argument inside the function, then you have to print it. Without the print statement, the output will not show the answer. So, it is important to capture the value the ***return*** gives us using a variable and to print it the output, we need to use ***print*** statement.

- Function should not do heavy-lifting. Keep it simple.

Returns

A function will take a number as input and will give output TRUE if the number is even else, False.

INPUT: **OUTPUT**:

```
def isNoEven (no):
    if no % 2 == 0:
        return True
    else:
        return False
res = isNoEven (100)
print (res)
```

TRUE

```
def isNoEven (no):
    if no % 2 == 0:
        print ("no is even")
        return True
    else:
        print ("no is odd")
```

FALSE

```
        return False

res = isNoEven (100)
print (res)

print (res)
```

Patterns

n = 5

*

* *

* * *

* * * *

* * * * *

n = 3

*

* *

* * *

n = 4

*

* *

* * *

* * * *

Whenever you multiply a string with an integer. You will repeat it that many times.

If n = 5 (it means 5 rows)

first row	➔	*
second row	➔	* *
third row	➔	* * *
fourth row	➔	* * * *
fifth row	➔	* * * * *

```
for row in range (1, n + 1):
        for noofstars in range (row):
                print (“*”, end=” ”) → this will
        print () → this will add a new line.
```

When row is 1, the program will print a *. After this, to go to new line, we will add print () so that a new line is added.

Now for row is 2, the for loop will work for 2 times, so it will first print a * and then will repeat the same for another *. Then it will go out of the loop and print ().

This will continue for row 3, 4 & 5.

INPUT:

```
print ("please enter no. of rows", end=" ")
noOfRows = int (input ())
for row in range (1, noOfRows + 1):
    for noofstars in range (row):
        print ("* ", end="")
    print ()
```

1st method – using for loop

```
def printPatternWithForLoop (rows):
    for row in range (1, rows + 1):
        for noOfStars in range (row):
            print ("*", end=" ")
        print ()
```

2nd Method – using while loop

```
def printPatternWithWhileLoop (rows):
    row = 1
    while row <= rows:
        noOfStars = 0
        while noOfStars < row:
            print ("*", end=" ")
            noOfStars += 1
        print ()
        row += 1
```

Whenever you multiply a string with an integer it will repeat it that many times.

3rd Method -

```
def printWithMultiplication (rows):
    for row in range (1, rows + 1):
        print ("* " * row)
```

```python
print ("Please enter no of rows", end="")
noOfRows = int (input ())

print ("With for loop")
printPatternWithForLoop (noOfRows)

print ("With while loop")
printPatternWithWhileLoop (noOfRows)

print ("With multipliation")
printWithMultiplication (noOfRows)
```