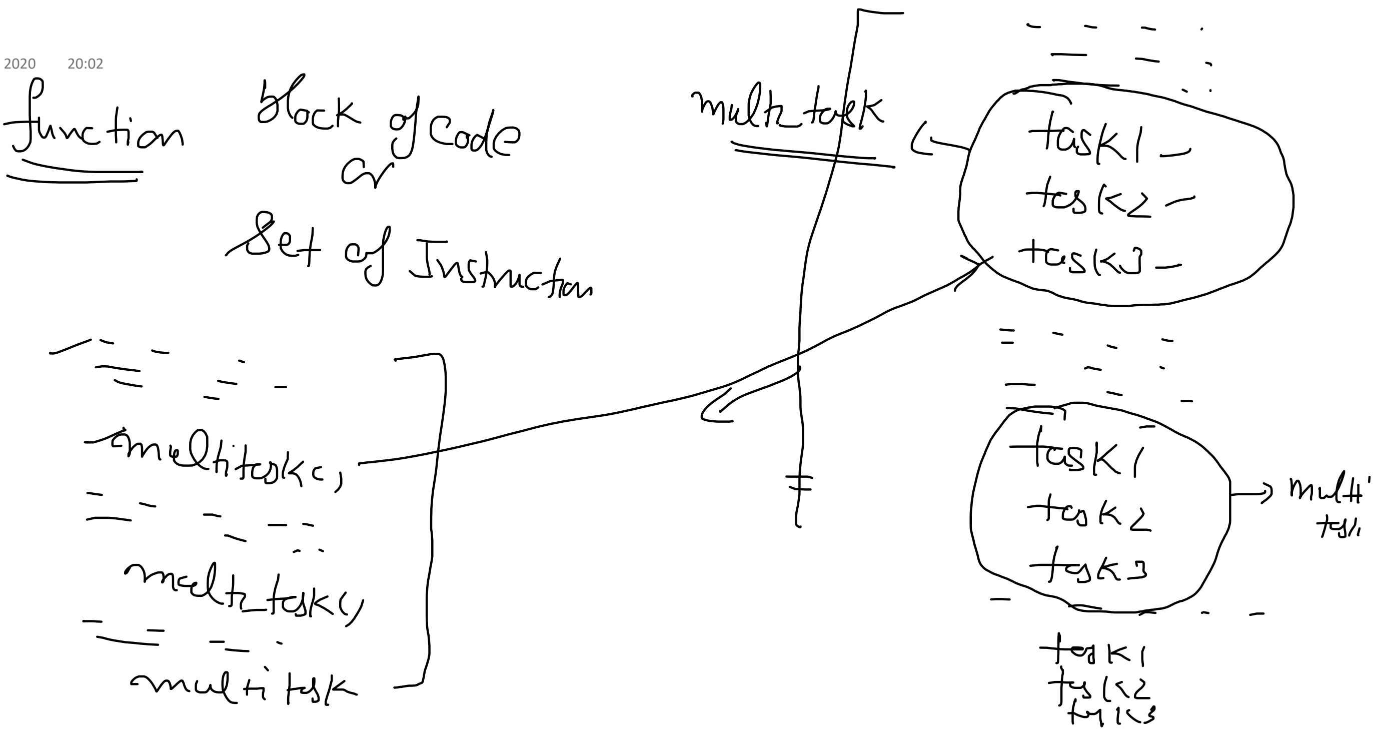
function block of Code
or
Set of Instruction



function:- it is a block of Code
(Set of Instruction)
(Set of operation / statement)

- It may/may not take additional Information
- It may / may not return data/values

l = [1, 2, 1, 3, 1, 1]

l.Count(1) → 4

l.append(5)

print(l.Count(1))
⇓
4!

print(l.append(5
⇓
None

Type I → function with no additional data or no return value

def function_name( ) : → definition
↓
Keyword
-print() } → Implementation
-if-else }
-for - -

=def func(): } → to do
pass

Type-I def func1():
print("this is my 1st function")

~~Type-II~~ func1() → Calling function

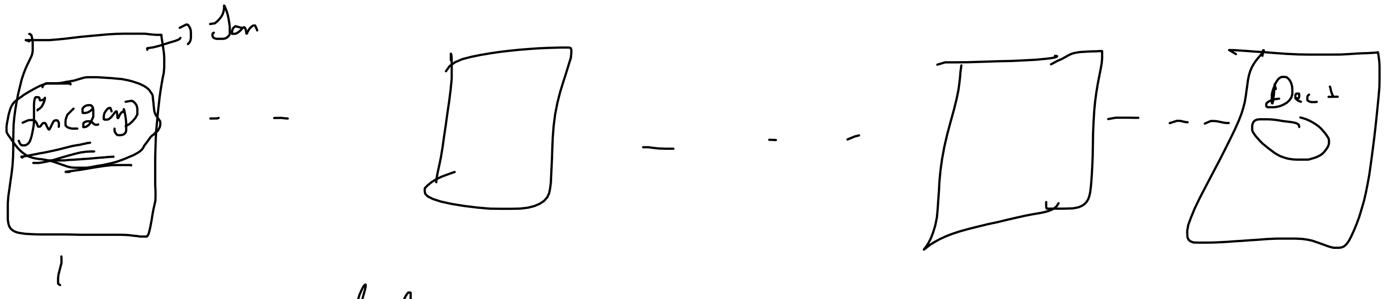
this is my 1st function

Type-2 taking additional data, no return
↳ argument

def func2(name):
print("Hello ", name)

func2('Varun') ⇒ Hello Varun

type :-



add *

if we don't know how many arguments that will be passed unto our function, add * before argument name

(it will take ~~values~~ as in tuple form

```
def  func4(* name):
        fname = name[0]
        lname = name[1]
        print(fname)
        print(lname)
```

("ravi", "Kumar")
0 1

```
func4("ravi", "Kumar")
```

when we are parsing multiple values, and we don't know how many argument it need so it make it generic we ~~v~~add * before argument

when we are parsing multiple key-value. and we don't know how many arguments it needs so it make a generic function add ** before argument name

County = ~~India~~ , India

```
open_account( fname, lastname, County )

//open_account('Varun', 'Kumar') ✗

open_account( fname, lastname, County="India")
```