→ functions

- how python reads your code
- Introduction to Functions
- diff arguments
- Scoping
- ⋮

27 lines along

Errors
- Compile time error

compiler is a program that reads your whole code at once & if there are no error it executes

compiler to convert your code into machine readable format

Programming language

They use an interpreter which reads code line by line and the first line when it sees an error it stops there

- compiled → C, C++
- interpreted → Ruby, Python
- hybrid (compiled + interpreted) → Java

→ How Python reads your code ??

→ we will later discuss this in depth

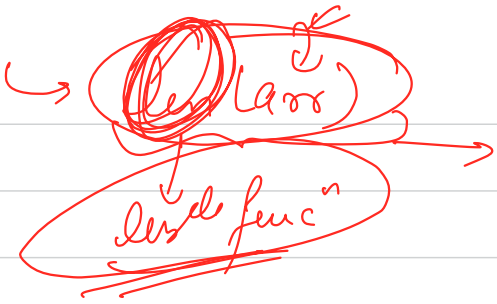
→ 1) Python is an interpreted language.

2) Python's interpreter reads our code line by line.

Functions → DRY (Don't Repeat Yourself)

→ club a piece of reusable unit of code & place somewhere else

→ function is a block of organised, reusable code that is used to perform single, related action to improve code modularity



arr = [- - -]

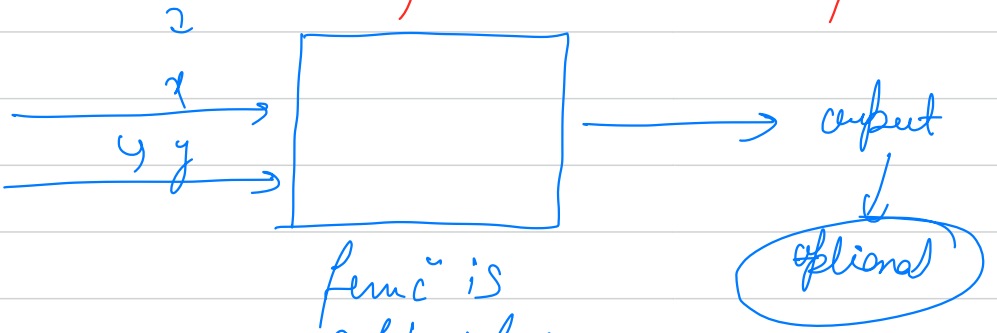
what if we have to work it?:

We have based on implementable 2 type func^n

1) In built func^n

2) User-defined

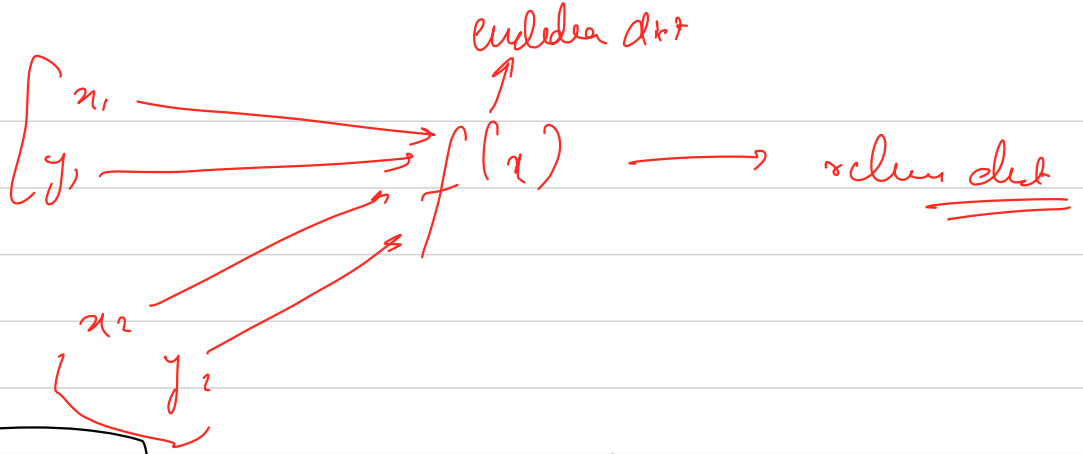
In maths also we have functions → $f(x)$



$f(x)$

$f(x, y)$

In python also, a function can or cannot take an input (argument), do some custom computation and may or may not return an output.



docstrings → Custom documentation that you can get for a func". We can write the name of a func" & get its docstring

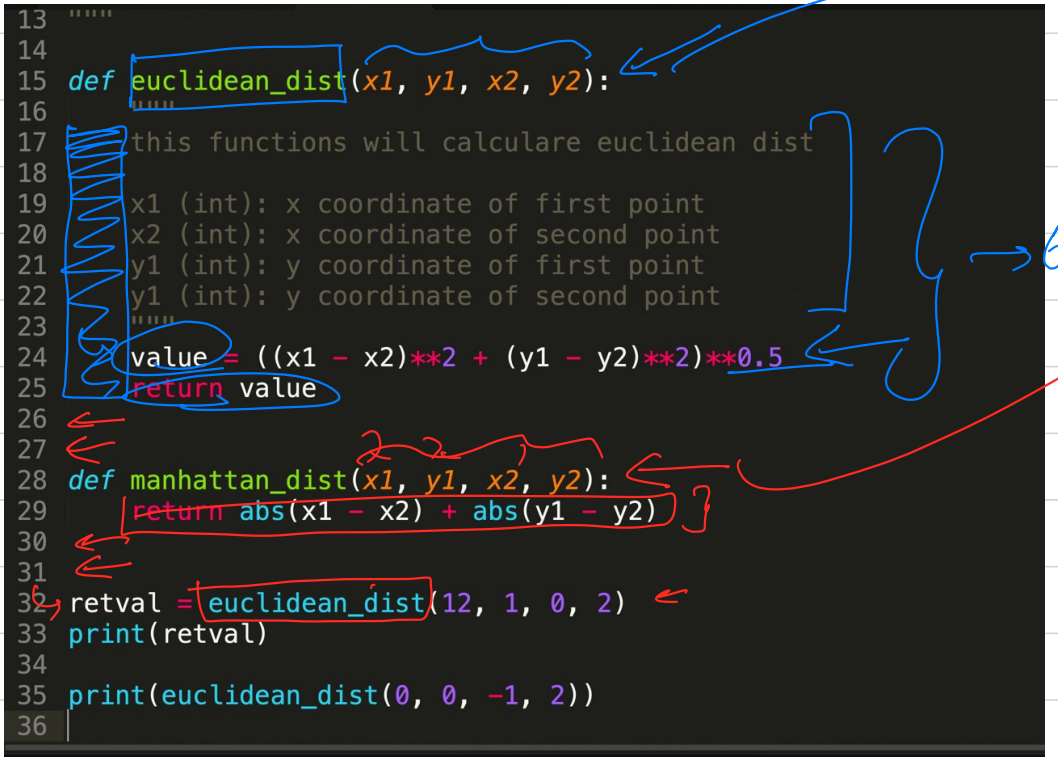
the return keyword

why we need it when we have print() ???

→ print is also a func°. It prints the value on the screen.

But let's assume if we want to use the output of our function later in the code.

if we want to use it, we will return it as a output from func° & store in a variable from where we have called.


"""
def euclidean_dist(x1, y1, x2, y2):
"""
this functions will calculare euclidean dist
x1 (int): x coordinate of first point
x2 (int): x coordinate of second point
y1 (int): y coordinate of first point
y1 (int): y coordinate of second point
"""
value = ((x1 - x2)**2 + (y1 - y2)**2)**0.5
return value
def manhattan_dist(x1, y1, x2, y2):
return abs(x1 - x2) + abs(y1 - y2)
retval = euclidean_dist(12, 1, 0, 2)
print(retval)
print(euclidean_dist(0, 0, -1, 2))

Arguments of a function

→ Argument is the input provided by user to func^{n}

- → Required arguments
- → keyword arguments
- → default arguments
- → variable - length arguments

Required arguments → These are arguments passed to a f^n in correct positional order. The no. of arguments passed while calling a function should be equal to no. of arguments defined in a func^n

Keyword arguments → These are used when calling a fem: You pass the parameter name with the value you want to use.

```
def fun(a, b):
    print(a + b)

fun(b=10, a=3)

fun(a=20, b=5)
```

Default argument → By using a default argument, we put a default value to a parameter while defining a function.

While making a function call, if we don't pass that parameter, then it's default value is used.

```
fun (a=10, b=20) {
    print(a+b)
}
fun(2, 3)
fun()
```

→ If you want to pass default arguments, then you should not have any required arguments to the right of all of them

variable length argument:

↳ We may want to process many inputs, but we don't know how many. Then we use variable length args.

Scope of a variable → scope defines visibility of a variable.

visibility → It means where can you use a variable.

→ local → when a variable is first defined inside a func it is only locally visible in the func.

→ global → visible everywhere until we have same variable in local scope