Functions

A fn is a reusable code that performs a specific task.

Instead of writing the same code again & again, your put it inside a fn & call it whenever needed.

Why do we use function?

- to avoid repeating code —increase code reusability.
- to reduce error & to reuse logic —
- to make debugging easier.

Types

Builtin – already present in python

eg: input(), print(), lower(), len()

User-Defined – fn which are created by programmer using def keyword.

lambda fn – anonymous, single life fns.

created with lambda keyword.

eg: sq = lambda a : a*a

print(sq(5))

Syntax: lambda argument: expre

eg: add = lambda a,b : a+b
print (add(1,2)) .

parameter — variable in parenthesis inside f^n definition

arguments — the actual value that is sent to f^n when it is called.

<u>Default parameter value.</u>

if a f^n called without an argument, it uses the default value.

eq:
```
def greet(name="minnu"):
    print ("hello", name)

greet()
greet("theertha")        → o/p   hello minnu
                                 hello theertha
```

<u>keyword arguments (~~kwargs~~) (kwargs)</u>

Your can send arguments with key=value syntax.

eq:
```
def fn(name, animal):
    print (name, "of my animal is ", animal)
fn(name="buddy", animal="cat")
```

Positional args → when you call a f^n with args without using keywords, they are called positional arguments.

eq:
```
def add(a,b):            → positional args must be in
    return a+b             correct order.
add (1,2)
```

Function

in pytho

we can

we can

we can

eq: de

de

ap

Functvon as Parametees.

in python fn is first class object

we can pass them to varvable.

we can pass them as parametees to another fns.

we can return them from fns.

eg:
```
def square (x):
        return x*x

    def apply-fn (fn-name, num):
            return fn-name(num)

    apply-fn ( square, 5)
```