Ch 8: Functions

What are functions?

→ A function is a code written by someone which can be used in the code. This helps in the reusability of code.

Types of functions:

User defined → created yourself
Built in → eg - print(), range(), len()

Syntax of creating function

```
def funcName(params):
    # Code here
```

We can wrap up large code inside a function. On using the function, the code inside it will get executed.

Use your function: funcName()

Example:

```
def myFunc():
    print("Code in function")
```

myFunc() → Output: Code in function

- Function with parameter

```
def paramFunc(a,b):
    print("Hello",(a*b))

paramFunc(2,3) → Hello 6
```

a b

```
n = paramFunc(6,7) ⟶ Hello 42
print(n)            None
```

↳ We get this because code returns None

- return statement

```
def func():
    return "Hello"

n = func()
print(n) ⟶ "Hello"
```

- Docstring

These are present in the first line of function to describe the function

```
def cube(n):
    ''' This function returns n^3
    Author: Unknown '''
```

Function with default parameter

```
def cube(n=2): # n will be 2 by default
    print(n**3)
```

cube(4) → 64
cube() → 8 (2^3)
cube(3) → 27

Recursion

Recursion is made for solving problems that can be broken into smaller repititive problems.

Iterative code of $\frac{n(n+1)}{2}$

```
def sum(n):
    sum = 0
    for i in range(n):
        sum = sum + (i+1)
    print(sum)
sum(3) → 6
```

Iterative code (using loops & conditions)

Functions are of 2 types (on the basis of code):

i. iterative
ii. recursive

Recursion is a function inside itself

```
def sum-recursive(n):
    if (n==0):
        return 0
    return n + sum-recursive(n-1)

print(sum-recursive(3)) → 6
```

sum_r(3) ⇌ 3 + sum_r(2) ⇌ 2 + sum_r(1) ⇌ 1 + sum_r(0) ⇌ sum_r(0) = 0

When program gets to know that sum_recursive(0) is 0, this loop is initiated in reverse.

- Naming Conventions :-

PascalCase → Used in classes

camelCase → used in variables/function

snake_case → Used in naming files.