Content Covered

- Concept
- Defining function
- Doc-String
- Closure
- Default Argument
- Arbitrary Argument
- Lambda function

Function

- For modularity, reuse & maintenance
- Has argument, code & return
- Pass is used for dummy place holder
- Syntax:

```
def function_name(parameters):
    --- statements ---
    return values
```

Example

```
def add(a, b, c):
    total = a + b + c
    return total

num_1 = 4
num_2 = 3
num_3 = 2
addition = add(num_1, num_2, num_3)
addition_2 = add(a=num_1, b=num_2, c=num_3)
print(addition)
print(addition_2)
```

Doc String

- Used for writing description of function and its arguments
- Description is inside triple quote
- Often handled by AI tools, we need to review

Variable Scope

- Global: Variable location and defined outside function
- Non-local: Variable located in parent function
- Local: Variable located inside a function
- Variable Preference: local → non – local → global
- We only get read access of variable defined outside function. To get write access we use **nonlocal** & **global**

Example

```
x = 10

def disp():
    x = 2
    print(f“From Function: {x}")

print(f“Before invocation {x}")
disp()
print(f“After invocation function {x}")
```

Default

- We can make argument of function optional by assigning default value
- For **mutable** data type assign **None** as default
- Example:

```
def increment(variable, incrementor=1):
    variable = variable + incrementor
    return variable
var = 5
var = increment(var)
var = increment(var, 5)
```

Args

- Used when n argument to a function. Example: sum
- All extra arguments are packed in args
- Example:

```
def productor(*args):
    product = 1
    for arg in args:
        product = product * arg
    return product
```

Kwargs

- Used when n named argument to a function.
- All extra name arguments are packed in kwargs
- Example:

```
def sql_generator(**kwargs):
    for key, value in kwargs.items():
        print(f"{key} = {value}")
```

Recursive Function

- A function that calls itself
- Solving task using recurrent relation but mayn't be optimal
- Example:

```
def factorial(n):
    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

Lambda

- One-line anonymous function without name
- Often used as helper function in map, reduce and filter
- Syntax:

  lambda input_arguments: output_expression

- Example:

```
sqr = lambda x: x ** 2
print(sqr(5))
```

Example

```
adder = lambda x, y: x + y
print(adder(5, 3))

even_chk = lambda x: 'Yes' if x % 2 == 0 else 'No'
print(even_chk(20))
```

Filter

- Used to filter out element in the list
- Used with lambda that yields True/False
- Example

```
input_age = [12, 18, 10, 26]
filtered_age = list(filter(lambda x:x >= 16, input_age))
print(filtered_age)
```

Map

- Used to apply common operation to all element in iterable
- Used with lambda that performs operation
- Example

```
txt = ['apple ', ' ball ']
clean_txt = list(map(lambda x:x.strip().upper(), txt))
print(clean_txt)
```

Exercise