FUNCTION, SCOPING AND ABSTRACTION (5 MARKS)

Why Function?

- Reusability
- concise code
- Modularity

Defining Functions

In [3]:
```
def greet():
    print("hello python")
```

In [4]:
```
print(greet())
```

```
hello python
None
```

Function specification

- It specifies the function name,parameters and the return type

In [ ]:
```
def add(x,y):
    """
    This function returns the addition of x and y
    perameters
    x (int) : the first number
    y (int) : the secound number

    Returns:
    int : The addition of x and y
    """
    result=x+y
    return result
```

In [ ]:
```
print(add(4,5))
```

```
9
```

In [ ]:
```
add(4,6)
```

Out[11]:
```
10
```

Types of Functions

- in built function
- user-defined functions

In [ ]:
```
dir(__builtins__) # A list cantaining inbuilt function
```

User defined functions

1. No parameters ,no returns

In [17]:
```
def printline():
    print("hello world")
printline()
```

```
hello world
```

2. No parameters , with return

In [19]:
```
def printline():
    return 'hello python'
printline()

```

Out[19]:
```
'hello python'
```

3. With parameters ,no returns

In [22]:
```
def printline(s):
    print(s)
m=input("Enter greeting ")
printline(m)
```

```
Enter greeting hello
hello
```

With parameters,with return

In [24]:
```
def printline(s):
    return s
    print("will this run") # This won't be execute

m=input("Enter greeting ")
printline(m)
```

```
Enter greeting hello
```

Out[24]:
```
'hello'
```

Returning multiple values

In [25]:
```
def sumsub(x,y):
    sum= x + y
    sub= x - y
    return sum,sub
```

In [27]:
```
m,n=sumsub(7,6)
print(type(m))
print(m)
print(type(n))
print(n)
```

```
<class 'int'>
13
<class 'int'>
1
```

In [28]:
```
a=7,6
type(a)
```

Out[28]:
```
tuple
```

In [29]:
```
a
```

Out[29]:
```
(7, 6)
```

WAP to return addition, substraction, multiplicatipon and division of given two numbers using function cal

In [30]:
```
def cal(x,y):
    return x+y,x-y,x*y,x/y

```

In [34]:
```
x=int(input("Enter first number "))
y=int(input("Enter secound number "))
add,sub,mul,div=cal(x,y)
print(f"addition        : {add}")
print(f"substraction    : {sub}")
print(f"multiplecation  : {mul}")
print(f"division        : {div}")
```

```
Enter first number1
Enter secound number2
addition        : 3
substraction    : -1
multiplecation  : 2
division        : 0.5
```

Parameters and arguments

- The values in perentheses used while defining function are called perameters
- The values passed while calling the function are the arguments

Types of arguments

Defult arguments

In [35]:
```
def square(x=20):
    return x*x
```

In [36]:
```
print(square())
print(square(10))
```

```
400
100
```

Positional arguments

- The number of arguments and their positions must watch
- If we change the order of arguments,the result will way
- If we change the number of arguments, we will get error

In [37]:
```
def sub(x,y):
    return x - y
```

In [ ]:
```
print(sub(5,6))
print(sub(6,5))
print(sub(5,6,7)) #Error
```

Keyword argument

- In case of all keyword arguments, the order does not matter.
- One can use combination of keyword and posiyional argument.
- Keyword argument always follows positional argument.

In [38]:
```
def wish(name,msg):
    print("hello ", name ,msg)
```

In [39]:
```
wish(name = 'pyhon',msg ='good morning')
```

```
hello  pyhon good morning
```

In [40]:
```
wish(msg ='good morning' , name = 'java')
```

```
hello  java good morning
```

In [41]:
```
wish("C++",msg = 'good morning')
```

```
hello  C++ good morning
```

In [43]:
```
wish(msg = 'good morning',"C++")
```

```
  File "<ipython-input-43-bb630304becd>", line 1
    wish(msg = 'good morning',"C++")
                              ^
SyntaxError: positional argument follows keyword argument
```

variable length arguments

In [44]:
```
def sum(*n):
    total = 0
    for i in n:
        total+=i
    print("The sum is ",total)
```

In [45]:
```
sum(10)
```

```
The sum is  10
```

In [46]:
```
sum(10,20)
```

```
The sum is  30
```

In [47]:
```
sum (10,-10,10,20)
```

```
The sum is  30
```

Function Scope

Local variable

- a local variable is declared when function has started execution and are lost when the function terminates

In [56]:
```
x=5
def fun():
    global x
    x=100
    print(x)
```

In [53]:
```
fun()
```

```
100
```

In [54]:
```
print(x)
```

```
100
```

Scoping rule

- `LEBG` Rule

  1.Local

  2.Enclosed

  3.Global

  4.Builtin

Nested Functions

In [57]:
```
def f():
    def g():
        print("Inside g function")
    print("inside f function")
```

In [60]:
```
f()
```

```
inside f function
```

In [61]:
```
def f():
    def g():
        print("Inside g function")
    g()
    print("inside f function")
```

In [62]:
```
f()
```

```
Inside g function
inside f function
```

In [64]:
```
def g(x):
    def h():
        x='abc'
        return x
    x=x+1
    print("in g function x is ",x)
    print(h())
    return x
```

In [65]:
```
x=3
z=g(x)
print(z)
```

```
in g function x is  4
abc
4
```

Duplicate Functions

In [71]:
```
def add(x,y):
    return x+y

def add(x,y):
    return x-y
```

In [72]:
```
print(add(8,6))
```

```
2
```

In [ ]:

In [ ]:

WAP to swap the first and last digit

```
 input = 123456
output = 623451
```

In [82]:

```
x=int(input("Enter number "))
temp=x
temp1=x
y=x%10
digit=0

while temp1!= 0:
    digit=digit+1
    temp1=temp1//10

while x>1:
    x=x//10
    z=x

num=temp//10

num=num*10+z

num1=num%10**(digit-1)

num1=y*10**(digit-1)+num1

print(num1)
```

```
Enter number 1234
4231
```

In [ ]:

```

```