Functions

- create a function with function name "add"
- function take two arguments a,b
- function must add a&b and print the sum of a and b if its number

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

Calling function

- Use function name to call the function and pass value 3 and 4 to arguments

```
add('x','y')
```

```
xy
```

Scope of Arguments

```
# global variable
c = 15

# function to perform addition
def add(a,b):
  c = 20
  output = a + b +c
  print(output)


# calling a function
add(2,3)
```

```
25
```

```
# global variable
a = "Global"

# function to prints global variables and local variables of function1
def function_1():
  f1 = "local_f1"
  print (f"{a} & {f}" )

# function 2 to prints global variablesand local variables of function1
def function_2():
  global f
  f = "local_f2"
  a = 3
  print(f"{a} & {f}" )


# calling a function
function_2()
function_1()
```

```
3 & local_f2
Global & local_f2
```

```
# global variable
a = "Global"


# function to prints global variables and local variables of function1
def outerFunction():
  o = 3
  def function_1():
    f = "local_f1"
    global f1 #global Keyword
    f1 = 1
    f2 =2
    print (f"{a} & {f}" )

# function 2 to prints global variablesand local variables of function1
def function_2():
  f = "local_f2"
  print (f"{a} & {f}" )
  print(f1)
  print(f2)


# calling a function
function_1()
function_2()
```

```
Global & local_f1
Global & local_f2
1
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-d9bae27af237> in <cell line: 23>()
     21 # calling a function
     22 function_1()
---> 23 function_2()

<ipython-input-3-d9bae27af237> in function_2()
     16   print (f"{a} & {f}" )
     17   print(f1)
---> 18   print(f2)
     19
     20

NameError: name 'f2' is not defined
```

Nested Functions & function stubs

```
def billing_system():
    cart = []

    def add_item(item_name, price):
        cart.append((item_name, price))

    def calculate_total():
        total = sum(price for _, price in cart)
        return total

    def apply_discount(total, discount_percentage):
        discount_amount = (discount_percentage / 100) * total
        total = total - discount_amount
        return total, discount_amount

    def generate_bill():
        print("Shopping Cart:")

        for item_name, price in cart:
```

```
            print(f"{item_name}: ${price:.2f}")
        total = calculate_total()

        discount_percentage = 10  # Example discount rate (10%)

        if len(cart) > 2:
            total_with_discount, discount_amount = apply_discount(total, discount_percentage)  # Capture the retu
            print(f"Total before discount: ${total:.2f}")
            print(f"Discount ({discount_percentage}%): ${discount_amount:.2f}")
            print(f"Total after discount: ${total_with_discount:.2f}")
        else:
            print(f"Total: ${total:.2f} (No discount applied)")

    while True:
        print("Options:")
        print("1. Add item to cart")
        print("2. Generate bill")
        print("3. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":
            item_name = input("Enter item name: ")
            price = float(input("Enter item price: "))
            add_item(item_name, price)
        elif choice == "2":
            generate_bill()
        elif choice == "3":
            break
        else:
            print("Invalid choice. Please try again.")

if __name__ == "__main__":
    billing_system()
```

```
Options:
1. Add item to cart
2. Generate bill
3. Exit
Enter your choice: 1
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-3-9d57a8158b61> in <cell line: 52>()
     51
     52 if __name__ == "__main__":
---> 53     billing_system()

                              2 frames
/usr/local/lib/python3.10/dist-packages/ipykernel/kernelbase.py in _input_request(self, prompt, ident,
parent, password)
    893             except KeyboardInterrupt:
    894                 # re-raise KeyboardInterrupt, to truncate traceback
--> 895                 raise KeyboardInterrupt("Interrupted by user") from None
    896             except Exception as e:
    897                 self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

```
def bill():
   def add_item(item_name, price):
        cart.append((item_name, price))

   def cal():
    print("Not implemented yet working on add-item")

   cal()



cart = []
item_name = input("Enter item name: ")
price = float(input("Enter item price: "))
add_item(item_name, price)

print(cart)
bill()
```

```
Enter item name: item1
Enter item price: 20
[('item1', 20.0)]
Not implemented yet working on add-item
```

Arguments

Keyword arguments and Default argument

```
def add(a,b,c=1): # c is default argument , a,b is keyword argument
  print(a+b+c)

add(a=3,b=4,c=5) # a = 3 , b = 4
```

```
12
```

Positional Arguments

```
def sub(a,b):
  print(a-b)

sub(3,4) # a = 3 , b = 4
```

```
-1
```

```
sub(4,3) # a = 4 , b = 3 , the order of the argument positions determines the values of arguments
```

```
1
```

Arbitrary arguments variable-length arguments

```
*args  - * Variable length argument without Keyword
**kwargs - ** Variable length argument with Keyword
```

```
def arbitrary_demo_args(*args):  # Variable length argument
    list_created = []  # Create an empty list
    for arg in args:
        list_created.append(arg)  # Append each argument to the list
    print(list_created)

arbitrary_demo_args('1', '6', '8')
```

```
['1', '6', '8']
```

```
def arbitrary_demo_kwargs(**a):  # Variable length argument with Keyword
    for key, val in a.items():
        print(f"Keyword: {key}, Value: {val}")

arbitrary_demo_kwargs(firstName='Aishwarya', secondName='Raj')
```

```
Keyword: firstName, Value: Aishwarya
Keyword: secondName, Value: Raj
```