Functions - I

🎯 Session Objectives

- ✅ Understand what functions are and why we use them.
- ✅ Learn to define functions, with parameters and arguments.
- ✅ Explore the use of the return statement.
- ✅ Understand scope and namespaces.

Types of Functions:

1. User-define Function:
   - Functions that we create with the help of def keyword and perform any custom task
2. Pre-Definded Function: (Built-In Function)
   - print()
   - len()
   - min()
   - max()
   - sum()
   - type()
   - input()

```
# Non Parameterized Function
def greet():
    print("Welcome to the Python Course!")

greet()
```

```
Welcome to the Python Course!
```

```
def Onboarding_Mail(name , designation , company = 'Ninjas'):
    print(f"Welcome to the Team, {name}.....")
    print(f"You are working as a {designation}, Please meet with your teammates.")
    print("Looking  forward to have a good association with you.")
    print(f"Thanks...")
    print(f"{company}")

Onboarding_Mail('Amit','Analyst','Amazon')
```

```
Welcome to the Team, Amit.....
You are working as a Analyst, Please meet with your teammates.
Looking  forward to have a good association with you.
Thanks...
Amazon
```

```
Onboarding_Mail('Utkarsh','Senior Analyst','American Express')
```

```
Welcome to the Team, Utkarsh.....
You are working as a Senior Analyst, Please meet with your teammates.
Looking  forward to have a good association with you.
Thanks...
American Express
```

```
Onboarding_Mail('Aditya','Lead Scientist')
```

```
Welcome to the Team, Aditya.....
You are working as a Lead Scientist, Please meet with your teammates.
Looking  forward to have a good association with you.
Thanks...
Ninjas
```

```python
def onboarding(employee_name, job_title , start_date, hr_name = "Bhupinder Jogi" ,
               company_name = "Coding Ninja" , address = "XYZ Location"):
    print(f"""
        Subject: Welcome to the Team!

            Dear {employee_name},

            We are excited to welcome you to {company_name}! On behalf of the entire team,
            we are thrilled to have you join us as {job_title}.

            Your onboarding will begin on {start_date}, and we have planned a smooth transition
            to help you settle into your new role. Please find the key details below:

            Start Date & Time: {start_date}

            Location / Login Details: {address}

            Point of Contact: "Devi Prashad Sharma"

            During your first week, you will be introduced to our team, guided through our systems,
            and provided with all the resources you need to excel. Our HR team will also walk you
            through our policies, benefits, and company culture.

            We believe your skills and experience will make a valuable contribution, and we look
            forward to seeing you thrive with us.

            Once again, welcome aboard and best wishes for a successful journey with {company_name}.

            Warm regards,
            {hr_name}
            HR Manager
            {company_name}
    """)
onboarding("Arpit","Senior Analyst","23rd Sept 2025" ,company_name = "PhonePe" )
```

Subject: Welcome to the Team!

Dear Arpit,

We are excited to welcome you to PhonePe! On behalf of the entire team,
we are thrilled to have you join us as Senior Analyst.

Your onboarding will begin on 23rd Sept 2025, and we have planned a smooth transition
to help you settle into your new role. Please find the key details below:

Start Date & Time: 23rd Sept 2025

Location / Login Details: XYZ Location

Point of Contact: "Devi Prashad Sharma"

During your first week, you will be introduced to our team, guided through our systems,
and provided with all the resources you need to excel. Our HR team will also walk you
through our policies, benefits, and company culture.

We believe your skills and experience will make a valuable contribution, and we look
forward to seeing you thrive with us.

Once again, welcome aboard and best wishes for a successful journey with PhonePe.

Warm regards,
Bhupinder Jogi
HR Manager
PhonePe

```python
# Arbitary arguments (*args)
employee_list = []
def employee_hired(*emp_name):
    return employee_list.append(emp_name)

emp_details = employee_hired('Shalu','Ali','Utkarsh','Aditya','Nihal','Kabiraj','Jagat')
type(emp_details)
```

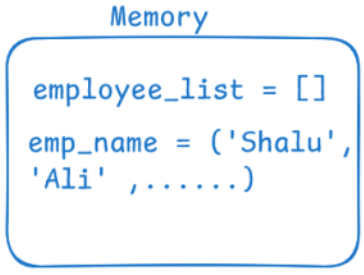
```
NoneType
```

```python
print(employee_list)
```

```
[('Shalu', 'Ali', 'Utkarsh', 'Aditya', 'Nihal', 'Kabiraj', 'Jagat')]
```

```python
a,b,*c = 1,2,3,4,5,6,7
print(a) # 1
print(b) # 2
print(c) # [3,4,5,6,7]
```

```
1
2
[3, 4, 5, 6, 7]
```



```python
# Arbitary arguments (*args)
employee_list = []
def employee_hired(*emp_name):
    for emp in emp_name:
        employee_list.append(emp)
    return employee_list
emp_details = employee_hired('Shalu','Ali','Utkarsh','Aditya','Nihal','Kabiraj','Jagat')
type(emp_details)
```

```
list
```

```python
print(employee_list)
```

```
['Shalu', 'Ali', 'Utkarsh', 'Aditya', 'Nihal', 'Kabiraj', 'Jagat']
```

```python
# Arbitary arguments (*args)
employee_list = []
def employee_hired(*emp_name):
    for emp in emp_name:
        employee_list.append(emp)
    return employee_list
emp_details = employee_hired('Shalu','Ali','Utkarsh','Aditya','Nihal','Kabiraj','Jagat')
type(emp_details)
```

```
list
```

```python
print(employee_list)
```

```
['Shalu', 'Ali', 'Utkarsh', 'Aditya', 'Nihal', 'Kabiraj', 'Jagat']
```

```python
a,b,*c = 1,2,3,4,5,6,7
print(a) # 1
print(b) # 2
print(c) # [3,4,5,6,7]
```

```
1
2
[3, 4, 5, 6, 7]
```

```python
def fav_car(car1, car2, car3, *car4): # (positional arg, arbitary argument)
    print(f"My Favourite Car is {car3}")

fav_car('Taigun','Slavia','Verna','Thar','Innova','Defender','Lord Alto',
            'Safari','Harrier','Bolero','XUV700','Altroz')
```

```
My Favourite Car is Verna
```

```python
# Keyword Argument
def fav_month(month1, month2, month3, month4 , month5):
    print("My Favourite Month" , month5)

fav_month(month2 = "Jan", month3 = "Oct", month4 = "Feb", month5 = "Dec", month1 = "July" )
```

```
My Favourite Month Dec
```

```python
# Positional Argument &  Keyword Argument
def fav_month(month1, month2, month3, month4 , month5):
    print("My Favourite Month" , month2)

fav_month("July","Jan", month3 = "Oct", month5 = "Dec", month4 = "Feb")
```

```
My Favourite Month Jan
```

```
SyntaxError: positional argument follows keyword argument
def fav_month(month1, month2, month3, month4 , month5):
    print("My Favourite Month" , month2)

fav_month(month3 = "Oct", month5 = "Dec", month4 = "Feb", "July","Jan")
```

```
# Default Parameter
def fav_car(car_name = "Safari"):
    print(car_name)

fav_car("Virtus")
```

```
Virtus
```

```
def fav_car(car_name = "Safari"):
    print(car_name)

fav_car()
```

```
Safari
```

```
def calculator(a,b,c,d=100,e=15):
    print((a*b)+ (c*d) + e)

calculator(10,20,c=5,e=20) # 720
```

```
720
```

```
car_list = ['Taigun','Slavia','Verna','Thar','Innova','Defender','Lord Alto',
            'Safari','Harrier','Bolero','XUV700','Altroz']
def iterative_car(cars):
    for car in cars:
        print(car , end = " - ")

iterative_car(car_list)
```

```
Taigun - Slavia - Verna - Thar - Innova - Defender - Lord Alto - Safari - Harrier - Bolero - XUV700 - Altroz -
```

```
def total_sum(val1, val2, val3, val4, val5):
    return val1 + val2 + val3 + val4 + val5

total = total_sum(100,200,300,400,500)
print(total)
```

```
1500
```

```
def total_sum(*val):
    return sum(val)

total = total_sum(100,200,300,400,500)
print(total)
```

```
1500
```

```python
def finding_max(*val):
    return max(val)

max_val = finding_max(11,22,33,44,55,1,2,4,5,7,4,99,111,4,5,9,50,66,81,91)
print(max_val)
```

```
111
```

```python
# Finding the maximum of 3 numbers
def max_three_number(x,y,z):
    if x>y and x>z:
        print(f"{x} is maximum")
    elif y>z and y>x:
        print(f"{y} is maximum")
    else:
        print(f"{z} is maximum")

max_three_number(15,20,11)
```

```
20 is maximum
```

```python
def max_three_number(x,y,z):
    if x>y and x>z:
        return x
    elif y>z and y>x:
        return y
    else:
        return z

max_ele = max_three_number(11,22,7) # 22
print(f"{max_ele} is the maximum element.")
```

```
22 is the maximum element.
```

```python
def max_three_number(x,y,z):
    return max(x,y,z)

max_ele = max_three_number(22,77,55) # 77
print(f"{max_ele} is the maximum element.")
```

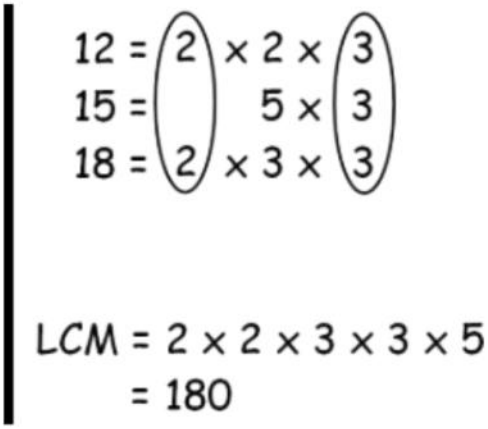
```
77 is the maximum element.
```

```python
# Calculate the area of a circle
pi = 3.14
def area_of_circle(radius):
    return pi * (radius ** 2)

area = area_of_circle(10)
print(area)
```

```
314.0
```

```python
# Area of Rectange
def area_of_rect(length , width):
    return length * width

rect_area = area_of_rect(10,20)
print(rect_area)
```

```
200
```




```python
import math
dir(math)
math.lcm(10,15)
```

```
30
```

```python
math.lcm(12,15,18)
```

```
180
```

```python
def celcius_to_fahrenheit(c):
    f = (1.8 * c) + 32
    return f

fahrenheit = celcius_to_fahrenheit(50)
print(fahrenheit)
```

```
122.0
```

```python
fahrenheit = celcius_to_fahrenheit(70)
print(fahrenheit)
```

```
158.0
```

```python
def fahrenheit_to_celcius(f):
    c = (f - 32)/1.8
    return c

celcius = fahrenheit_to_celcius(120)
print(celcius)
```

```
48.888888888888886
```

```python
# bill_amount > 1500 -> 25% discount else 10% discount
def billing_system(*item_cost):
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
    if bill_amount > 1500:
        final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else:
        final_amt_to_pay = bill_amount - (bill_amount * 0.10)
    return final_amt_to_pay
        # final_amt_to_pay = (bill_amount * 0.75)
total_pay = billing_system(100,200,300,400,500,600,700)
print(total_pay)
```

```
2100.0
```

```python
# Positional Argument with Arbitary Argument [Qty * item_cost]
def billing_system(qty,*item_cost):
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
    if bill_amount > 1500:
        final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else:
        final_amt_to_pay = bill_amount - (bill_amount * 0.10)
    return final_amt_to_pay * qty
        # final_amt_to_pay = (bill_amount * 0.75)
total_pay = billing_system(2,100,200,300,400,500,600,700)
print(total_pay)
```

```
4200.0
```

```python
# Positional Argument with Arbitary Argument [GST Addition] [5% GST Standard]
def billing_system(gst,*item_cost):
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
    if bill_amount > 1500:
        final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else:
        final_amt_to_pay = bill_amount - (bill_amount * 0.10)
    return final_amt_to_pay + (final_amt_to_pay * gst)
        # final_amt_to_pay = (bill_amount * 0.75)
total_pay = billing_system(0.05,100,200,300,400,500,600,700)
print(total_pay)
```

```
2205.0
```

```
gst_slab = {
    'grocery' : 0.05,
    'clothings' : 0.12,
    'shoes' : 0.18,
    'cab_service' : 0.05,
    'luxury_item' : 0.40
}
bill_invoice = {
    'grocery' : 100,
    'clothings' : 200,
    'shoes' : 300,
    'cab_service': 400,
    'luxury_item' : 700
}
```

```
bill_invoice.items()
```

```
dict_items([('grocery', 100), ('clothings', 200), ('shoes', 300), ('cab_service', 400), ('luxury_item', 700)])
```

```
total_bill = []
for gst_slab_items , gst_slab_gst in gst_slab.items():
    for bill_invoice_items , bill_invoice_amount in bill_invoice.items():
        if gst_slab_items == bill_invoice_items:
            bill_invoice_amount = bill_invoice_amount + (bill_invoice_amount * gst_slab_gst)
            total_bill.append(bill_invoice_amount)
        else:
            continue

print(total_bill)
```

```
[105.0, 224.0, 354.0, 420.0, 980.0]
```

```
# Arbitary Argument
def billing_system(*item_cost):
    bill_amount = 0
    for amt in item_cost:
        bill_amount += amt
    return bill_amount

total_pay = billing_system(*total_bill) # * with iterables will make it work
print(total_pay)
```

```
2083.0
```

```
gst_slab = {
    'grocery' : 0.05,
    'clothings' : 0.12,
    'shoes' : 0.18,
    'cab_service' : 0.05,
    'luxury_item' : 0.40
}
bill_invoice = {
    'grocery' : [100],
    'clothings' : [200],
    'shoes' : [300,500],
    'cab_service': [400],
    'luxury_item' : [600,700]
}
```

```
bill_invoice.values()
```

```
dict_values([[100], [200], [300, 500], [400], [600, 700]])
```

```
flatten_list = []
for row in bill_invoice.values():
    for val in row:
        flatten_list.append(val)

flatten_list
```

```
[100, 200, 300, 500, 400, 600, 700]
```

```
flatten_list = []
for row in bill_invoice.values():
    for val in row:
        flatten_list.append(val)

flatten_list
total_bill = []
for gst_slab_items , gst_slab_gst in gst_slab.items():
    for bill_invoice_items , bill_invoice_amount in bill_invoice.items():
        if gst_slab_items == bill_invoice_items:
            for val in bill_invoice_amount:
                bill_invoice_amount = val + (val * gst_slab_gst)
                total_bill.append(bill_invoice_amount)
        else:
            continue

print(total_bill)
```

```
[105.0, 224.0, 354.0, 590.0, 420.0, 840.0, 980.0]
```

```python
def billing_system(*item_cost):
    bill_amount = 0
    for amt in item_cost:
        bill_amount += amt
    return bill_amount

total_pay = billing_system(*total_bill) # * with iterables will make it work
print(total_pay)
```

```
3513.0
```