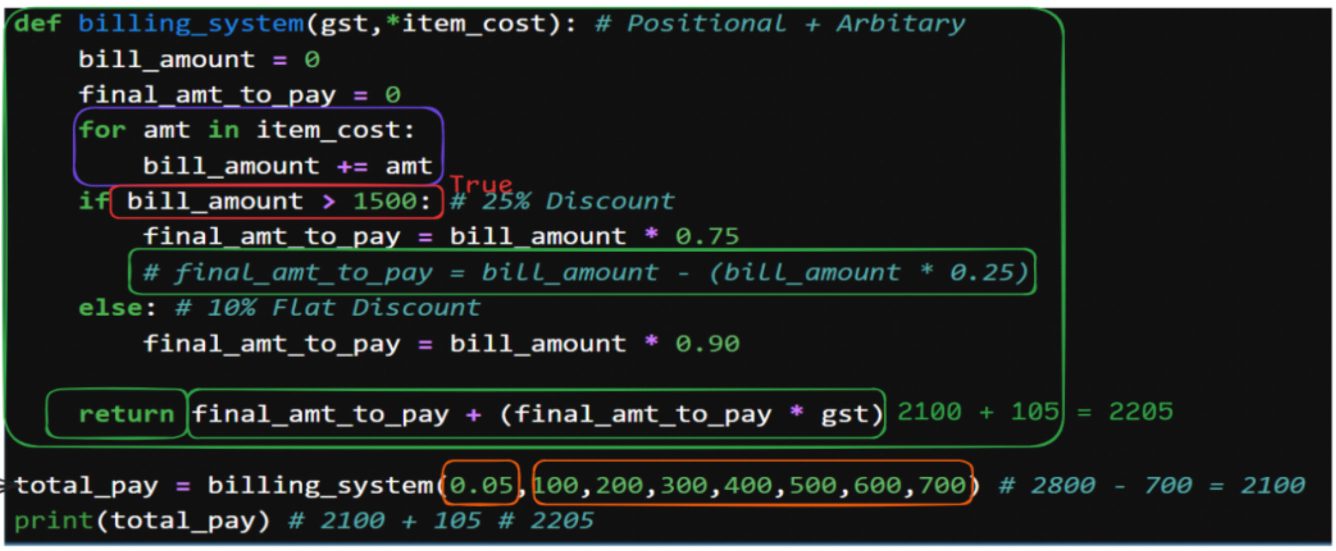
Functions - I

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.

Python Searches in LEGB Order:

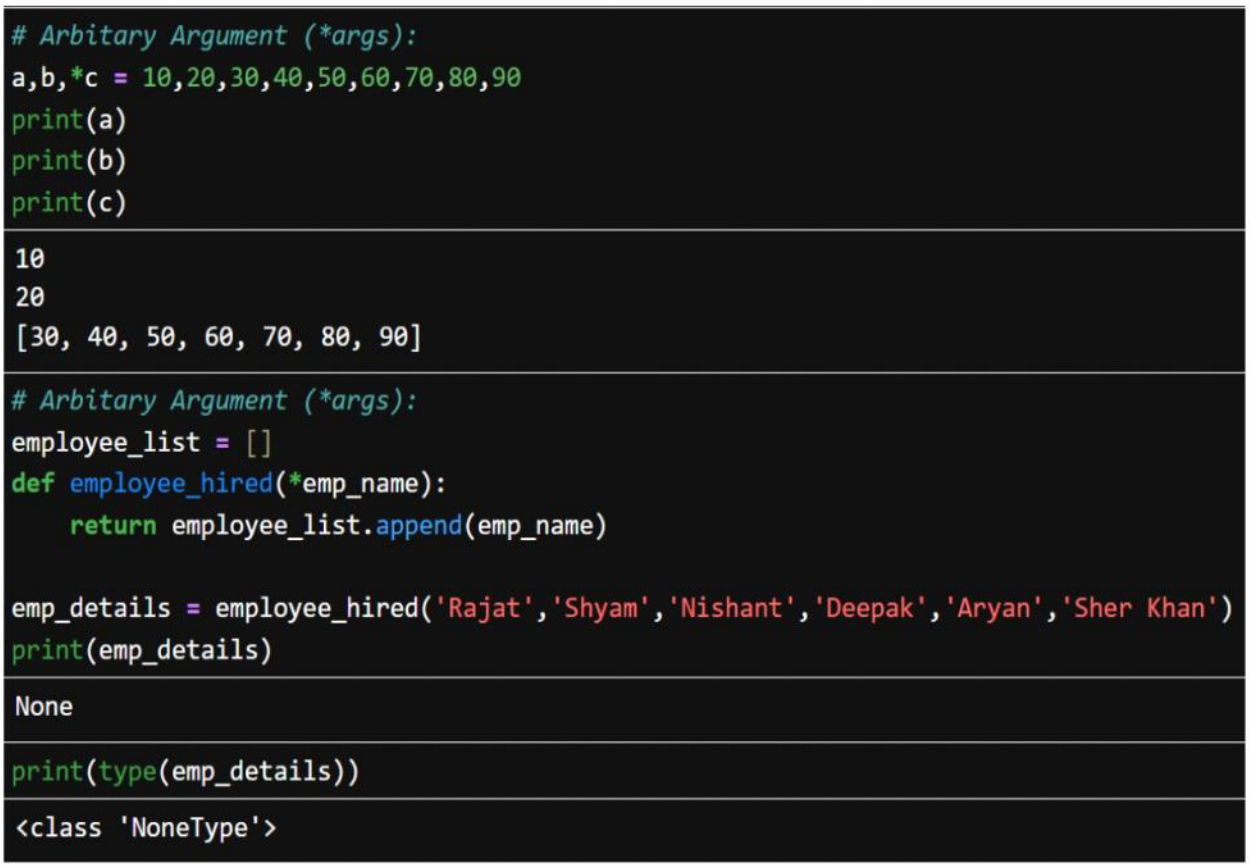
Level	Meaning
L	Local: inside current function
E	Enclosing: functions inside functions
G	Global: top-level script
B	Built-in: Python's built-in functions



```
def billing_system(gst,*item_cost): # Positional + Arbitary
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
                                       True
    if bill_amount > 1500: # 25% Discount
        final_amt_to_pay = bill_amount * 0.75
        # final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else: # 10% Flat Discount
        final_amt_to_pay = bill_amount * 0.90

    return final_amt_to_pay + (final_amt_to_pay * gst) 2100 + 105 = 2225

total_pay = billing_system(0.05,100,200,300,400,500,600,700) # 2800 - 700 = 2100
print(total_pay) # 2100 + 105 # 2205
```

Memory

```
total_pay = None

billing_system(0.05,100,200,300,400,500,600,700)
                                               amt
gst = ~~None~~ 0.05

item_cost = ~~None~~ (100,200,300,400,500,600,700)


billing_amount = 2800
final_amt_to_pay = 2800*.75 = 2100
= 2800 - 700 = 2100
```



```
# Arbitary Argument (*args):
a,b,*c = 10,20,30,40,50,60,70,80,90
print(a)
print(b)
print(c)
```

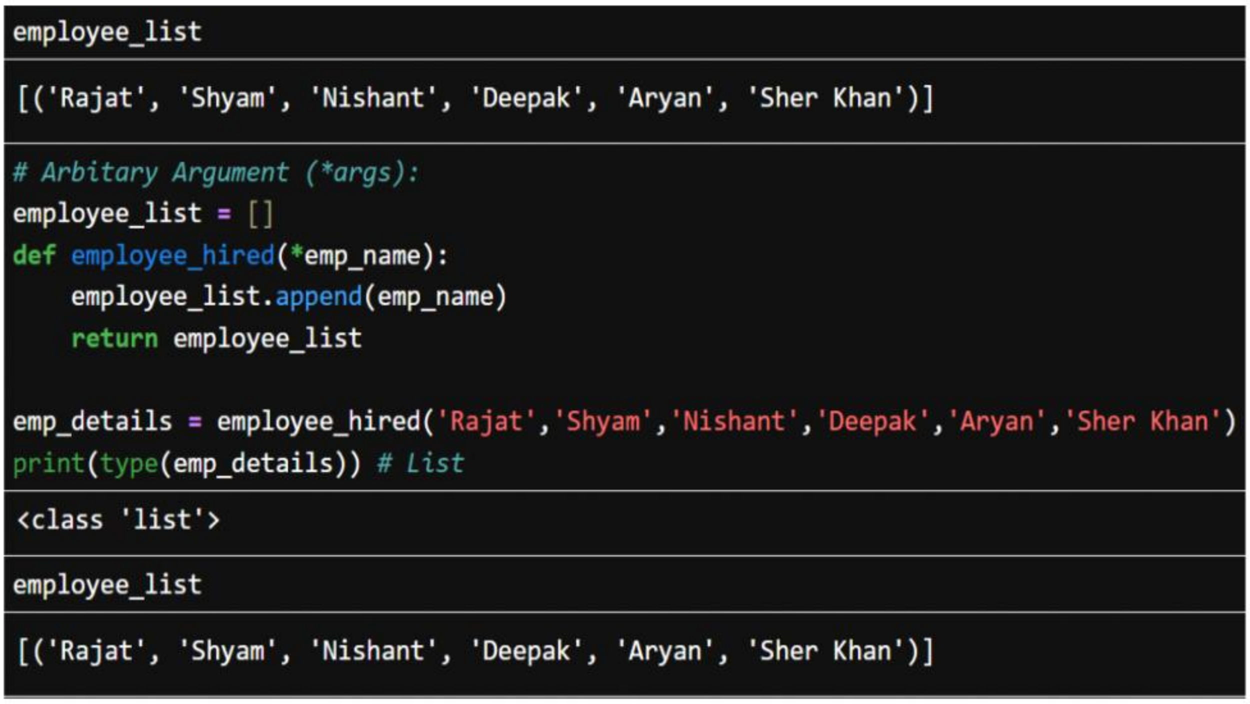
```
10
20
[30, 40, 50, 60, 70, 80, 90]
```

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

emp_details = employee_hired('Rajat','Shyam','Nishant','Deepak','Aryan','Sher Khan')
print(emp_details)
```

```
None
```

```
print(type(emp_details))
```

```
<class 'NoneType'>
```



```
employee_list
```

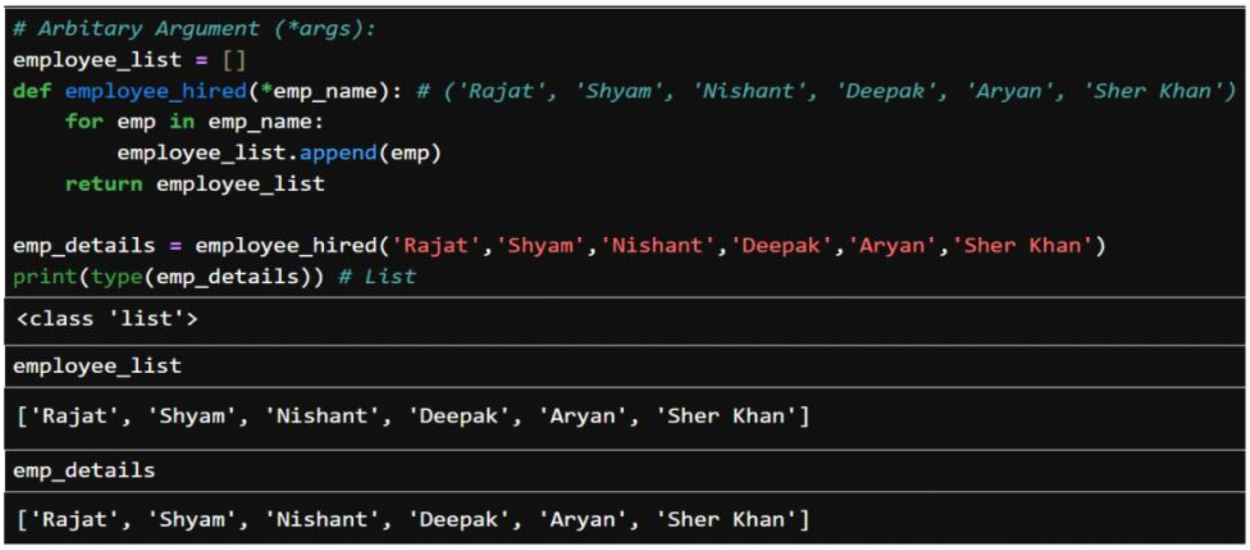
```
[('Rajat', 'Shyam', 'Nishant', 'Deepak', 'Aryan', 'Sher Khan')]
```

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

emp_details = employee_hired('Rajat','Shyam','Nishant','Deepak','Aryan','Sher Khan')
print(type(emp_details)) # List
```

```
<class 'list'>
```

```
employee_list
```

```
[('Rajat', 'Shyam', 'Nishant', 'Deepak', 'Aryan', 'Sher Khan')]
```



```
# Arbitary Argument (*args):
employee_list = []
def employee_hired(*emp_name): # ('Rajat', 'Shyam', 'Nishant', 'Deepak', 'Aryan', 'Sher Khan')
    for emp in emp_name:
        employee_list.append(emp)
    return employee_list

emp_details = employee_hired('Rajat','Shyam','Nishant','Deepak','Aryan','Sher Khan')
print(type(emp_details)) # List
```

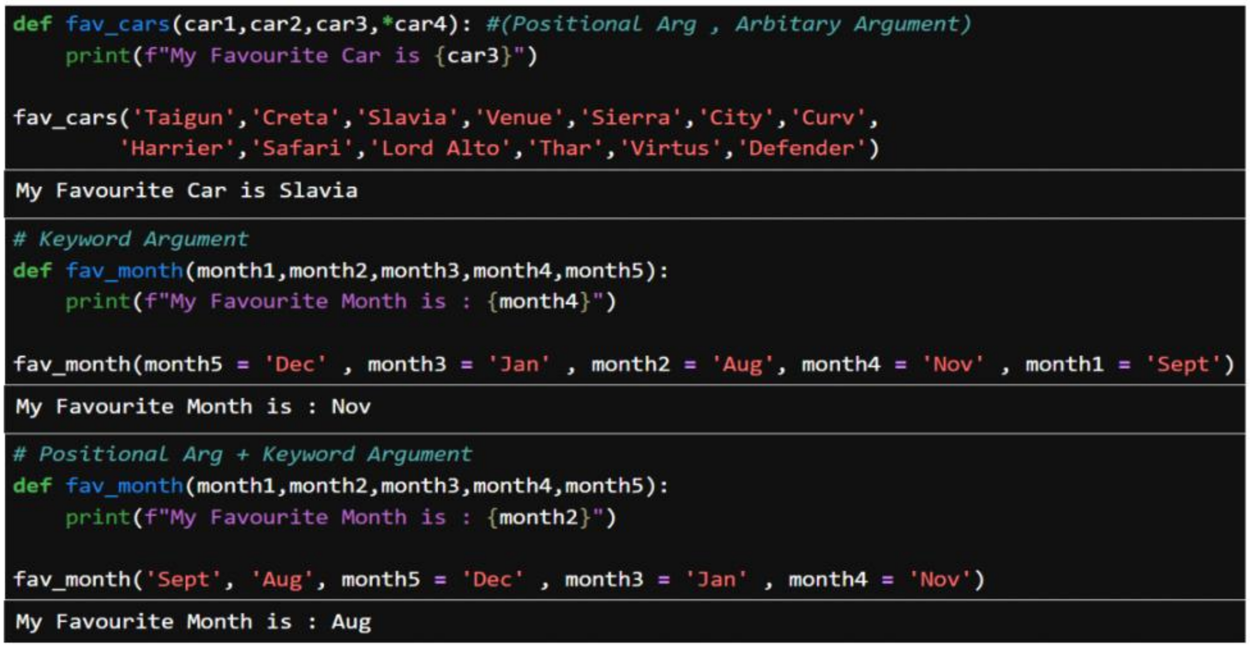
```
<class 'list'>
```

```
employee_list
```

```
['Rajat', 'Shyam', 'Nishant', 'Deepak', 'Aryan', 'Sher Khan']
```

```
emp_details
```

```
['Rajat', 'Shyam', 'Nishant', 'Deepak', 'Aryan', 'Sher Khan']
```



```
def fav_cars(car1,car2,car3,*car4): #(Positional Arg , Arbitary Argument)
    print(f"My Favourite Car is {car3}")

fav_cars('Taigun','Creta','Slavia','Venue','Sierra','City','Curv',
        'Harrier','Safari','Lord Alto','Thar','Virtus','Defender')
```

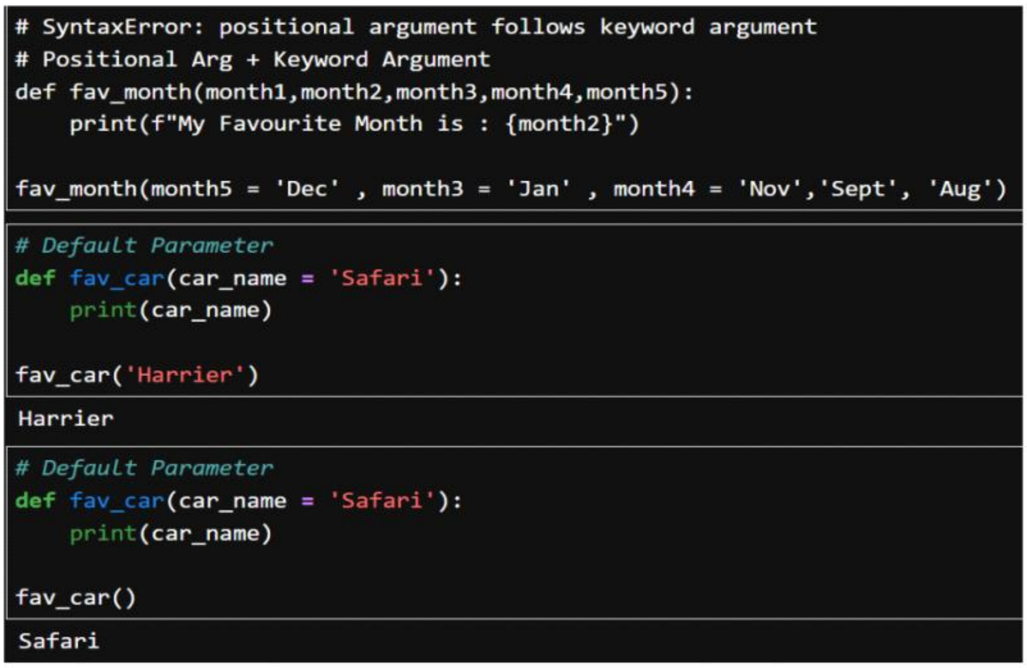
```
My Favourite Car is Slavia
```

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

fav_month(month5 = 'Dec' , month3 = 'Jan' , month2 = 'Aug', month4 = 'Nov' , month1 = 'Sept')
```

```
My Favourite Month is : Nov
```

```
# Positional Arg + Keyword Argument
def fav_month(month1,month2,month3,month4,month5):
    print(f"My Favourite Month is : {month2}")

fav_month('Sept', 'Aug', month5 = 'Dec' , month3 = 'Jan' , month4 = 'Nov')
```

```
My Favourite Month is : Aug
```



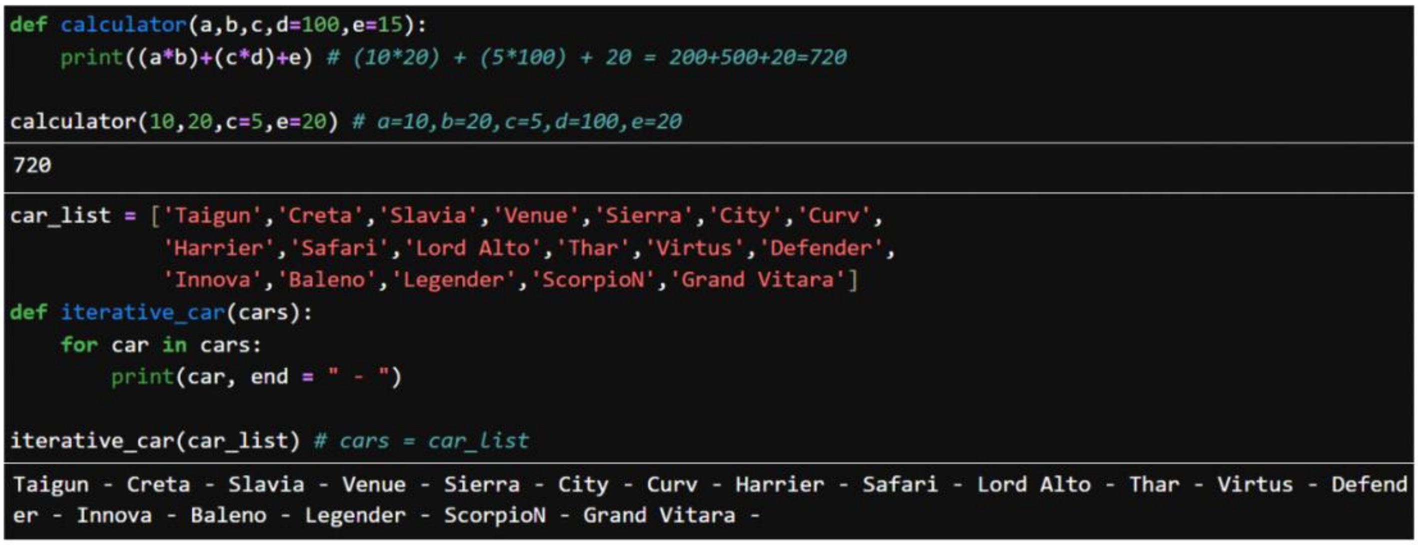
```
# SyntaxError: positional argument follows keyword argument
# Positional Arg + Keyword Argument
def fav_month(month1,month2,month3,month4,month5):
    print(f"My Favourite Month is : {month2}")

fav_month(month5 = 'Dec' , month3 = 'Jan' , month4 = 'Nov','Sept', 'Aug')
```

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

fav_car('Harrier')
```

```
Harrier
```

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

fav_car()
```

```
Safari
```

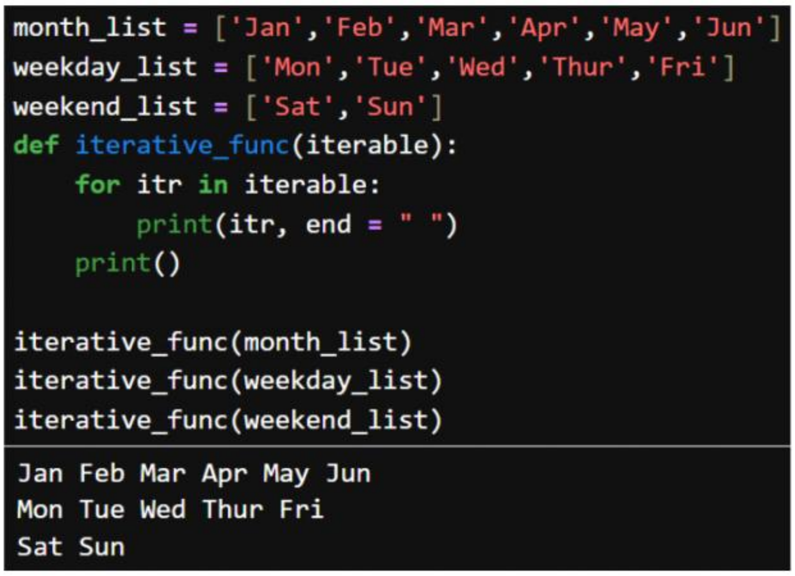


```python
def calculator(a,b,c,d=100,e=15):
    print((a*b)+(c*d)+e) # (10*20) + (5*100) + 20 = 200+500+20=720

calculator(10,20,c=5,e=20) # a=10,b=20,c=5,d=100,e=20
```

```
720
```

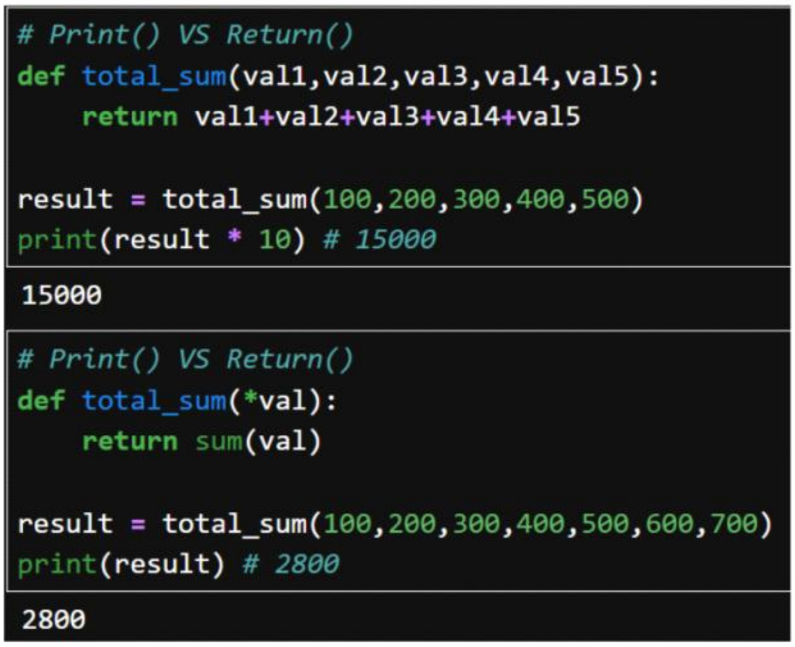
```python
car_list = ['Taigun','Creta','Slavia','Venue','Sierra','City','Curv',
            'Harrier','Safari','Lord Alto','Thar','Virtus','Defender',
            'Innova','Baleno','Legender','ScorpioN','Grand Vitara']
def iterative_car(cars):
    for car in cars:
        print(car, end = " - ")

iterative_car(car_list) # cars = car_list
```

```
Taigun - Creta - Slavia - Venue - Sierra - City - Curv - Harrier - Safari - Lord Alto - Thar - Virtus - Defend
er - Innova - Baleno - Legender - ScorpioN - Grand Vitara -
```



```python
month_list = ['Jan','Feb','Mar','Apr','May','Jun']
weekday_list = ['Mon','Tue','Wed','Thur','Fri']
weekend_list = ['Sat','Sun']
def iterative_func(iterable):
    for itr in iterable:
        print(itr, end = " ")
    print()

iterative_func(month_list)
iterative_func(weekday_list)
iterative_func(weekend_list)
```

```
Jan Feb Mar Apr May Jun
Mon Tue Wed Thur Fri
Sat Sun
```

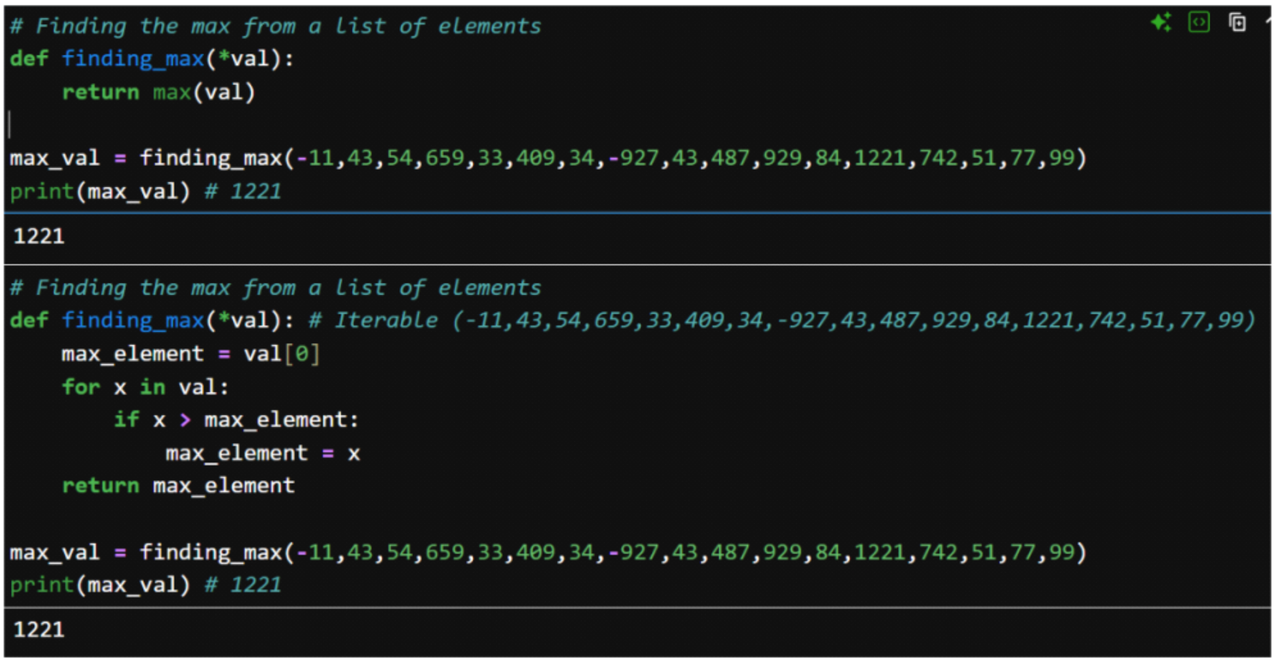


```python
# Print() VS Return()
def total_sum(val1,val2,val3,val4,val5):
    return val1+val2+val3+val4+val5

result = total_sum(100,200,300,400,500)
print(result * 10) # 15000
```

```
15000
```

```python
# Print() VS Return()
def total_sum(*val):
    return sum(val)

result = total_sum(100,200,300,400,500,600,700)
print(result) # 2800
```

```
2800
```

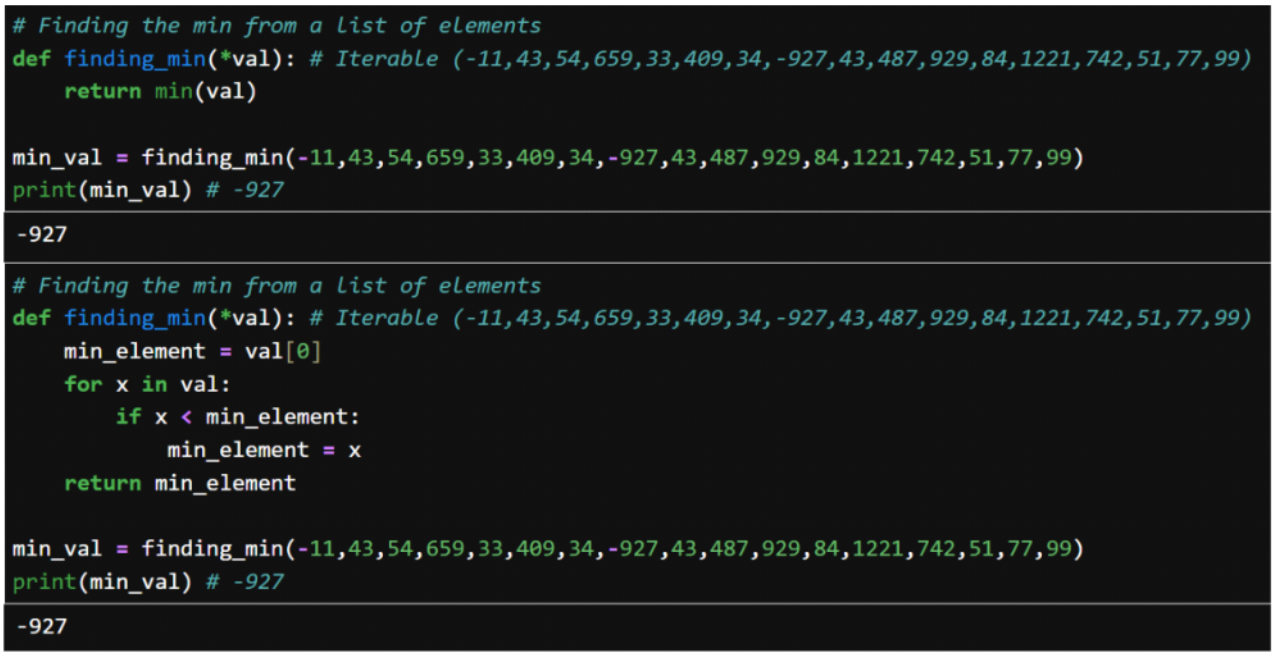


```python
# Finding the max from a list of elements
def finding_max(*val):
    return max(val)

max_val = finding_max(-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
print(max_val) # 1221
```

```
1221
```

```python
# Finding the max from a list of elements
def finding_max(*val): # Iterable (-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
    max_element = val[0]
    for x in val:
        if x > max_element:
            max_element = x
    return max_element

max_val = finding_max(-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
print(max_val) # 1221
```

```
1221
```

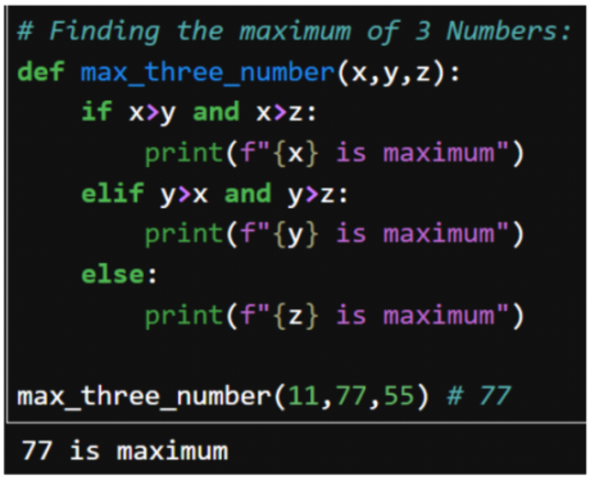


```python
# Finding the min from a list of elements
def finding_min(*val): # Iterable (-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
    return min(val)

min_val = finding_min(-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
print(min_val) # -927
```

```
-927
```

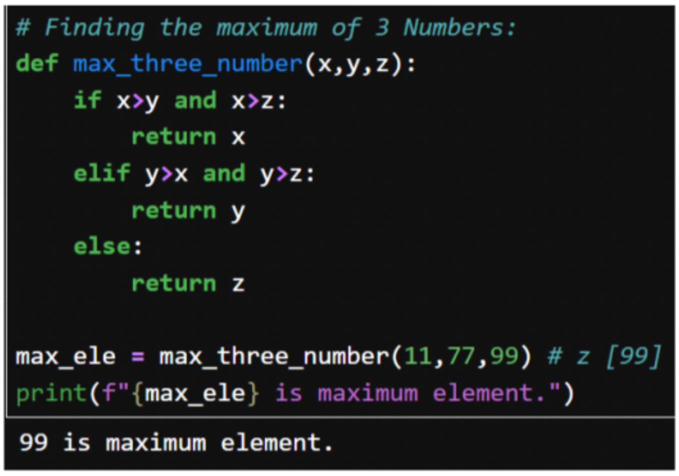
```python
# Finding the min from a list of elements
def finding_min(*val): # Iterable (-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
    min_element = val[0]
    for x in val:
        if x < min_element:
            min_element = x
    return min_element

min_val = finding_min(-11,43,54,659,33,409,34,-927,43,487,929,84,1221,742,51,77,99)
print(min_val) # -927
```

```
-927
```



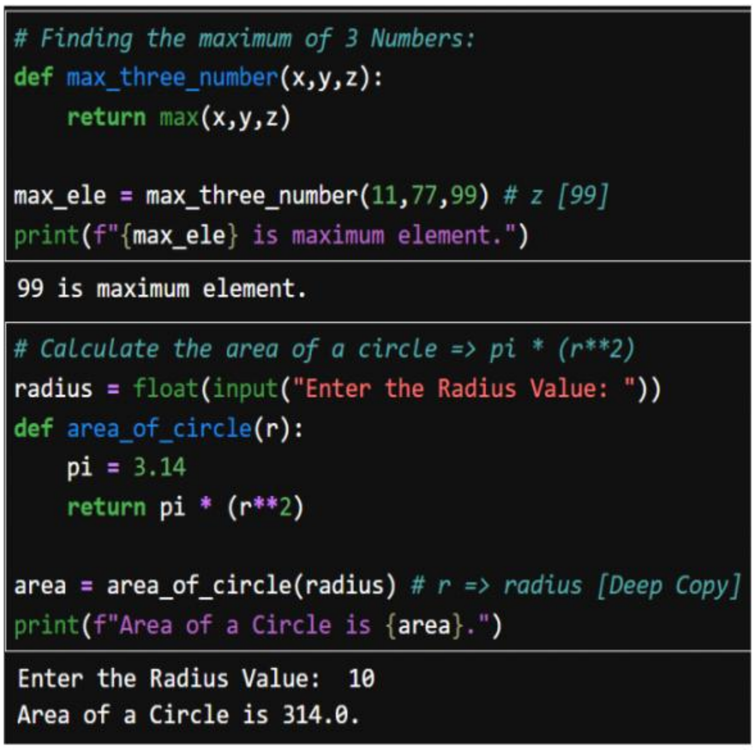
```python
# Finding the maximum of 3 Numbers:
def max_three_number(x,y,z):
    if x>y and x>z:
        print(f"{x} is maximum")
    elif y>x and y>z:
        print(f"{y} is maximum")
    else:
        print(f"{z} is maximum")

max_three_number(11,77,55) # 77
```

```
77 is maximum
```



```python
# Finding the maximum of 3 Numbers:
def max_three_number(x,y,z):
    if x>y and x>z:
        return x
    elif y>x and y>z:
        return y
    else:
        return z

max_ele = max_three_number(11,77,99) # z [99]
print(f"{max_ele} is maximum element.")
```

```
99 is maximum element.
```

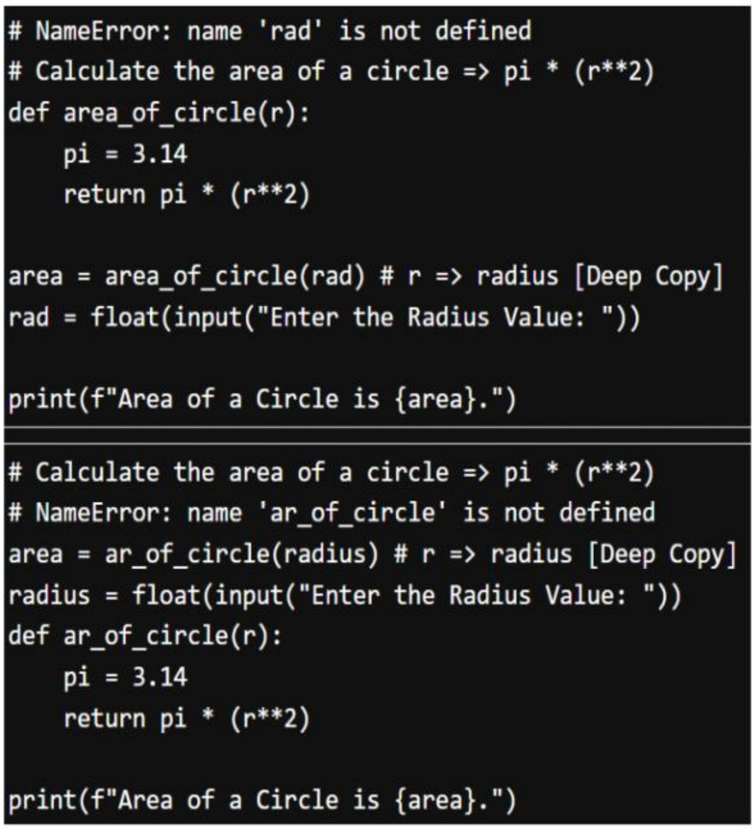


```python
# Finding the maximum of 3 Numbers:
def max_three_number(x,y,z):
    return max(x,y,z)

max_ele = max_three_number(11,77,99) # z [99]
print(f"{max_ele} is maximum element.")
```

```
99 is maximum element.
```

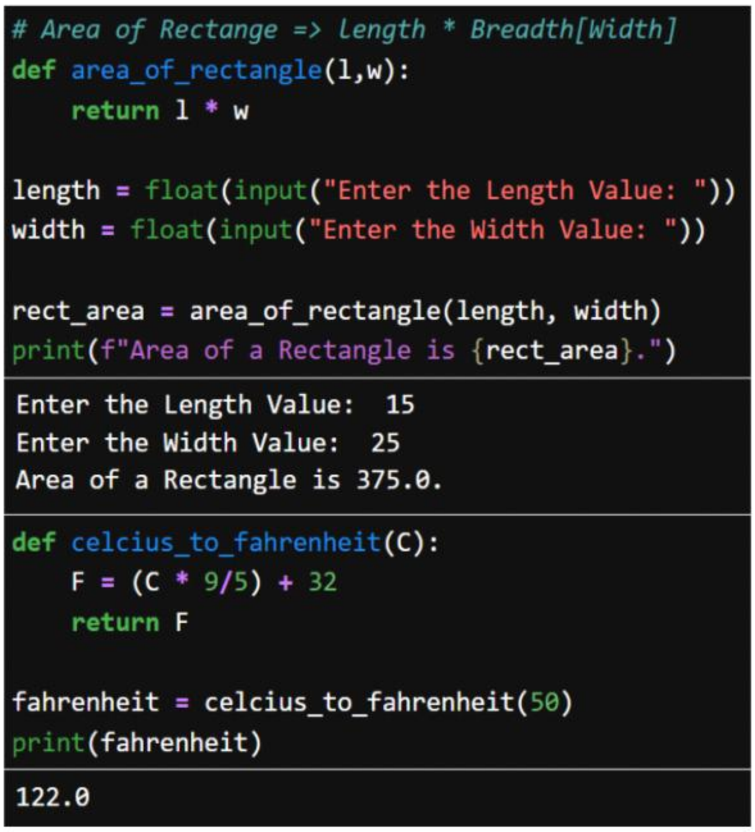
```python
# Calculate the area of a circle => pi * (r**2)
radius = float(input("Enter the Radius Value: "))
def area_of_circle(r):
    pi = 3.14
    return pi * (r**2)

area = area_of_circle(radius) # r => radius [Deep Copy]
print(f"Area of a Circle is {area}.")
```

```
Enter the Radius Value:  10
Area of a Circle is 314.0.
```



```python
# NameError: name 'rad' is not defined
# Calculate the area of a circle => pi * (r**2)
def area_of_circle(r):
    pi = 3.14
    return pi * (r**2)

area = area_of_circle(rad) # r => radius [Deep Copy]
rad = float(input("Enter the Radius Value: "))

print(f"Area of a Circle is {area}.")
```

```python
# Calculate the area of a circle => pi * (r**2)
# NameError: name 'ar_of_circle' is not defined
area = ar_of_circle(radius) # r => radius [Deep Copy]
radius = float(input("Enter the Radius Value: "))
def ar_of_circle(r):
    pi = 3.14
    return pi * (r**2)

print(f"Area of a Circle is {area}.")
```

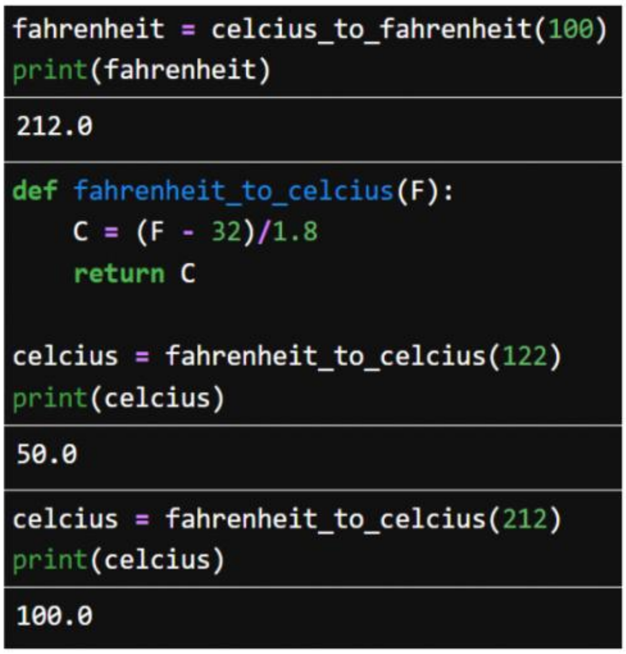


```python
# Area of Rectange => Length * Breadth[Width]
def area_of_rectangle(l,w):
    return l * w

length = float(input("Enter the Length Value: "))
width = float(input("Enter the Width Value: "))

rect_area = area_of_rectangle(length, width)
print(f"Area of a Rectangle is {rect_area}.")
```

```
Enter the Length Value:  15
Enter the Width Value:  25
Area of a Rectangle is 375.0.
```

```python
def celcius_to_fahrenheit(C):
    F = (C * 9/5) + 32
    return F

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

```
122.0
```



```python
fahrenheit = celcius_to_fahrenheit(100)
print(fahrenheit)
```

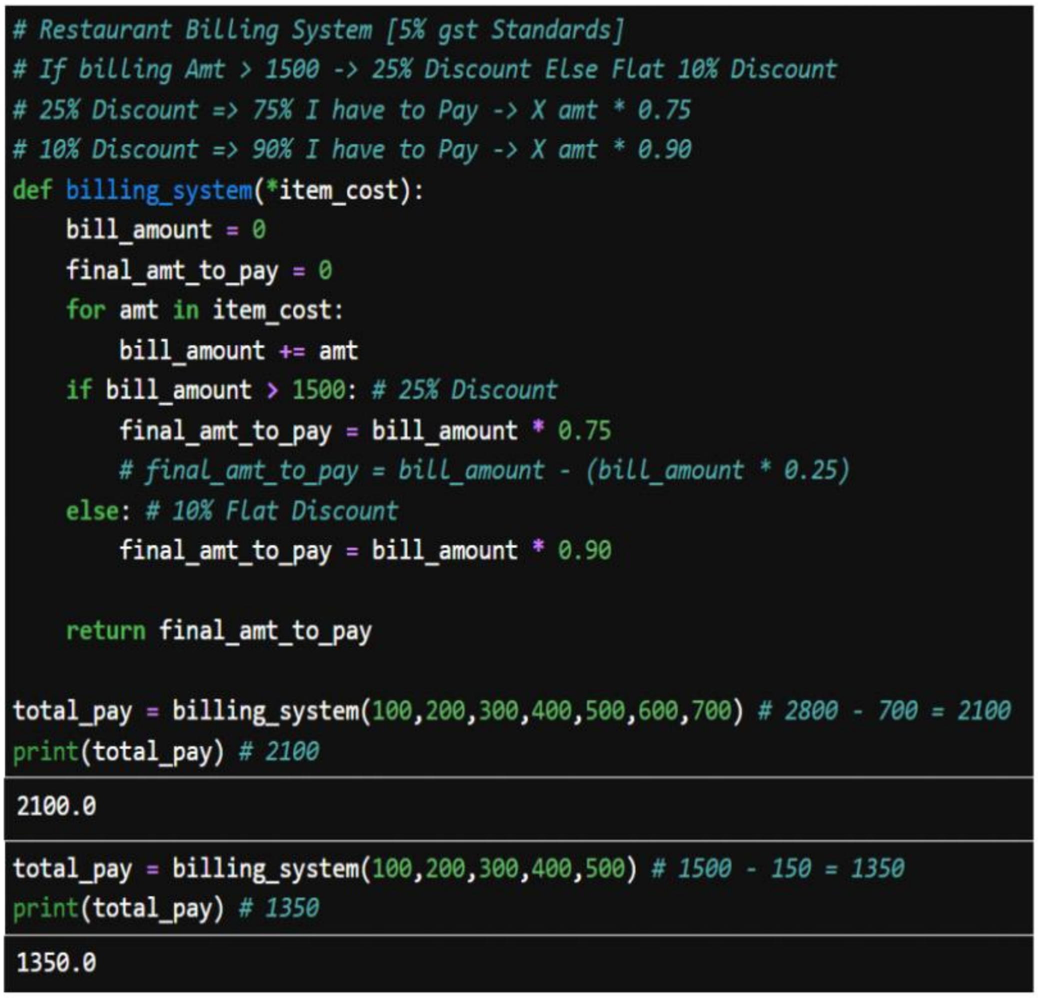
```
212.0
```

```python
def fahrenheit_to_celcius(F):
    C = (F - 32)/1.8
    return C

celcius = fahrenheit_to_celcius(122)
print(celcius)
```

```
50.0
```

```python
celcius = fahrenheit_to_celcius(212)
print(celcius)
```

```
100.0
```

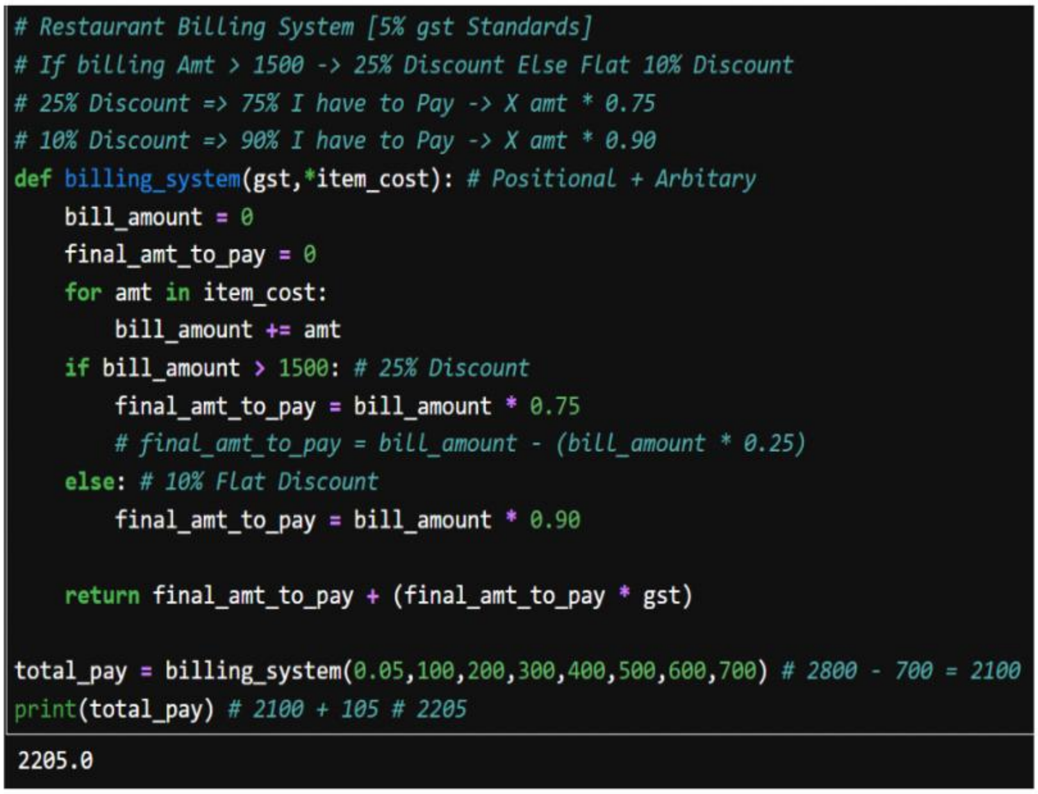


```python
# Restaurant Billing System [5% gst Standards]
# If billing Amt > 1500 -> 25% Discount Else Flat 10% Discount
# 25% Discount => 75% I have to Pay -> X amt * 0.75
# 10% Discount => 90% I have to Pay -> X amt * 0.90
def billing_system(*item_cost):
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
    if bill_amount > 1500: # 25% Discount
        final_amt_to_pay = bill_amount * 0.75
        # final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else: # 10% Flat Discount
        final_amt_to_pay = bill_amount * 0.90

    return final_amt_to_pay

total_pay = billing_system(100,200,300,400,500,600,700) # 2800 - 700 = 2100
print(total_pay) # 2100
```

```
2100.0
```

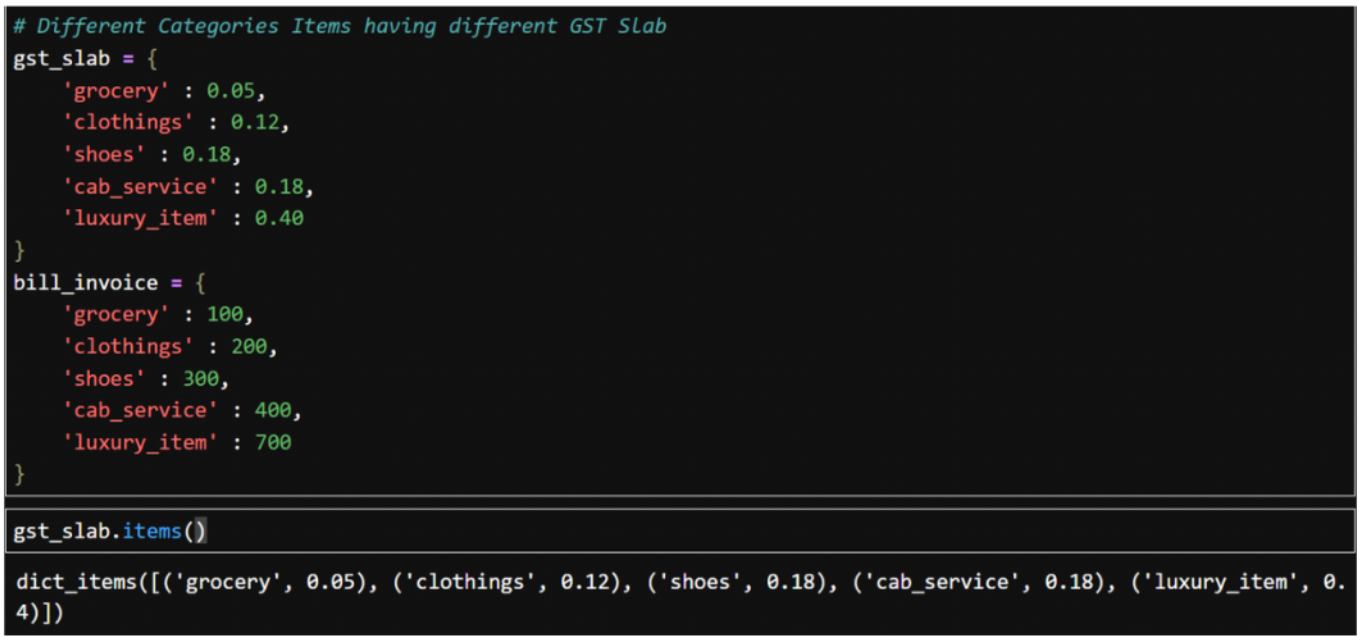
```python
total_pay = billing_system(100,200,300,400,500) # 1500 - 150 = 1350
print(total_pay) # 1350
```

```
1350.0
```



```python
# Restaurant Billing System [5% gst Standards]
# If billing Amt > 1500 -> 25% Discount Else Flat 10% Discount
# 25% Discount => 75% I have to Pay -> X amt * 0.75
# 10% Discount => 90% I have to Pay -> X amt * 0.90
def billing_system(gst,*item_cost): # Positional + Arbitary
    bill_amount = 0
    final_amt_to_pay = 0
    for amt in item_cost:
        bill_amount += amt
    if bill_amount > 1500: # 25% Discount
        final_amt_to_pay = bill_amount * 0.75
        # final_amt_to_pay = bill_amount - (bill_amount * 0.25)
    else: # 10% Flat Discount
        final_amt_to_pay = bill_amount * 0.90

    return final_amt_to_pay + (final_amt_to_pay * gst)

total_pay = billing_system(0.05,100,200,300,400,500,600,700) # 2800 - 700 = 2100
print(total_pay) # 2100 + 105 # 2205
```

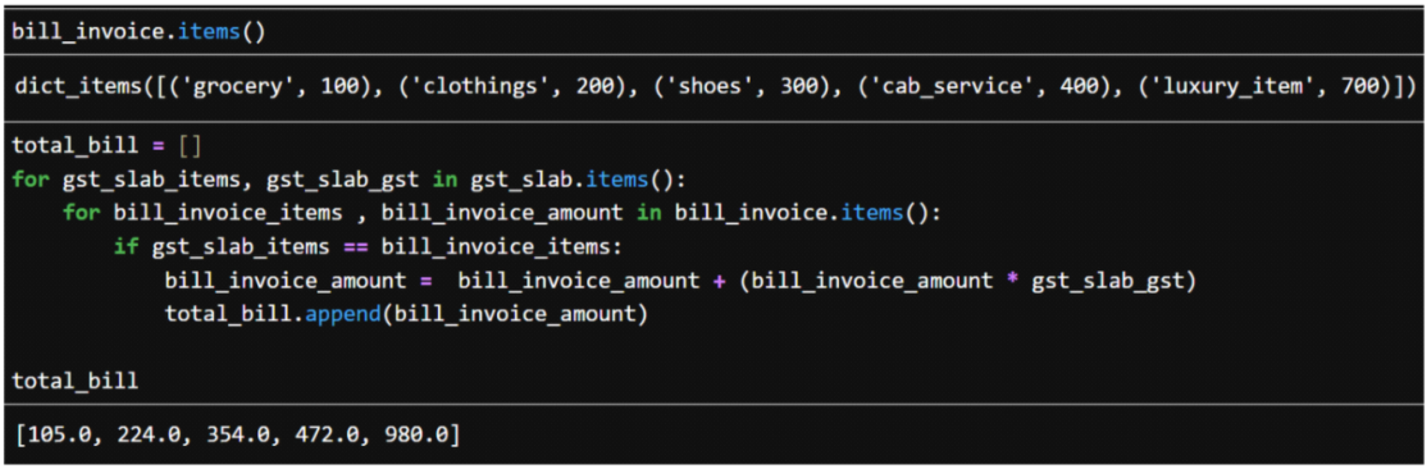
```
2205.0
```



```python
# Different Categories Items having different GST Slab
gst_slab = {
    'grocery' : 0.05,
    'clothings' : 0.12,
    'shoes' : 0.18,
    'cab_service' : 0.18,
    'luxury_item' : 0.40
}
bill_invoice = {
    'grocery' : 100,
    'clothings' : 200,
    'shoes' : 300,
    'cab_service' : 400,
    'luxury_item' : 700
}
```

```python
gst_slab.items()
```

```
dict_items([('grocery', 0.05), ('clothings', 0.12), ('shoes', 0.18), ('cab_service', 0.18), ('luxury_item', 0.
4)])
```

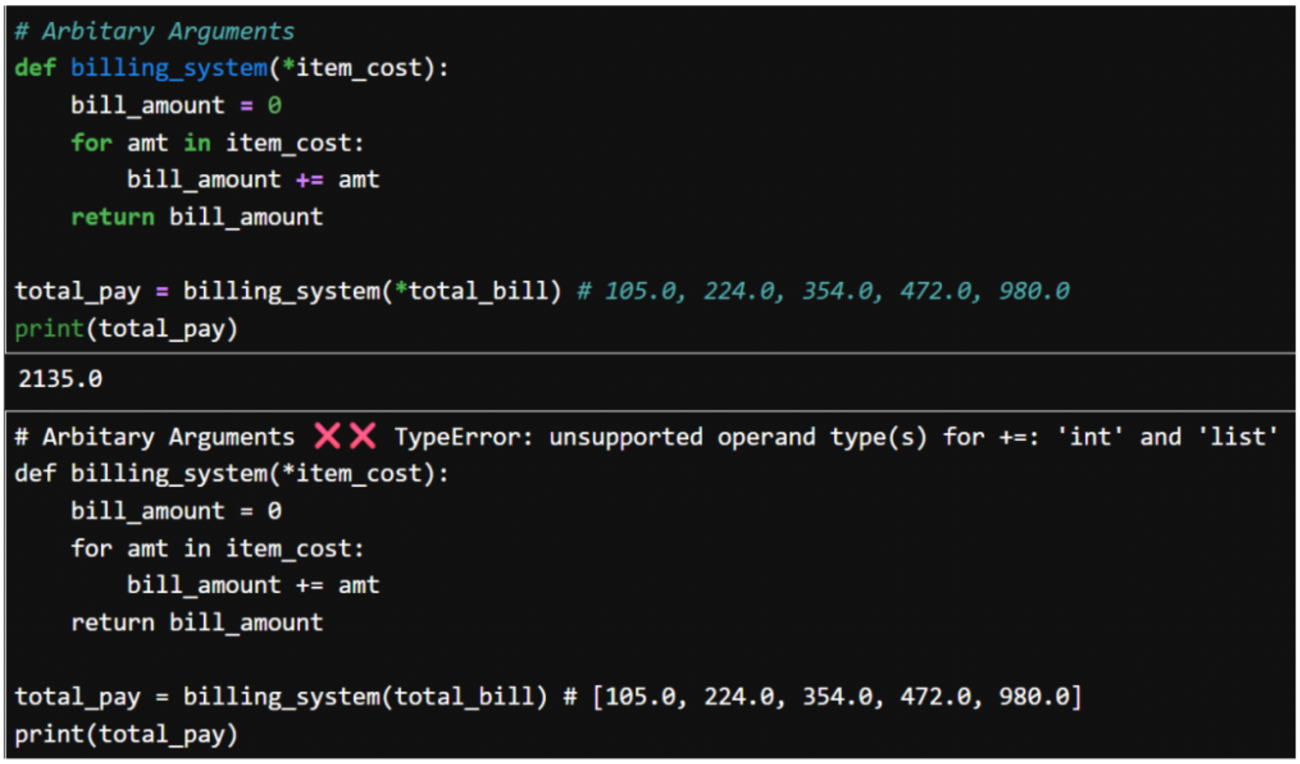


```python
bill_invoice.items()
```

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

```python
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)

total_bill
```

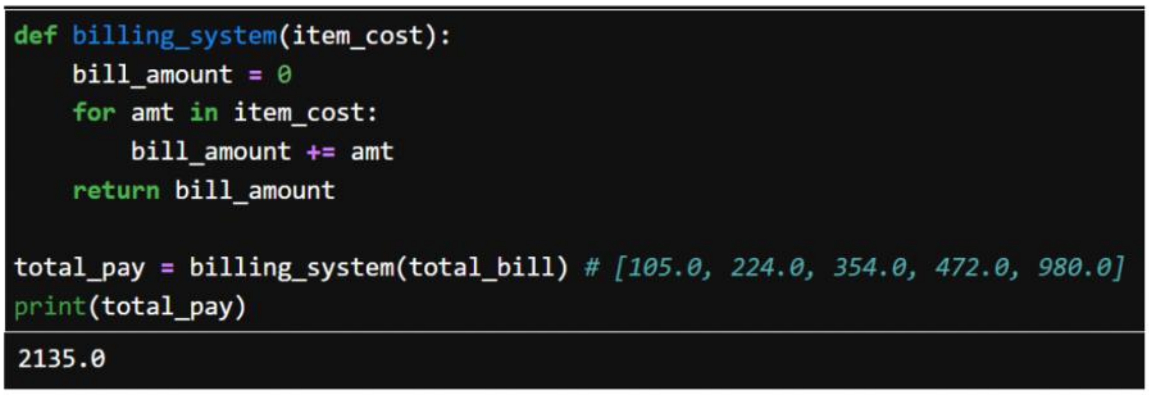
```
[105.0, 224.0, 354.0, 472.0, 980.0]
```



```python
# Arbitary Arguments
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) # 105.0, 224.0, 354.0, 472.0, 980.0
print(total_pay)
```

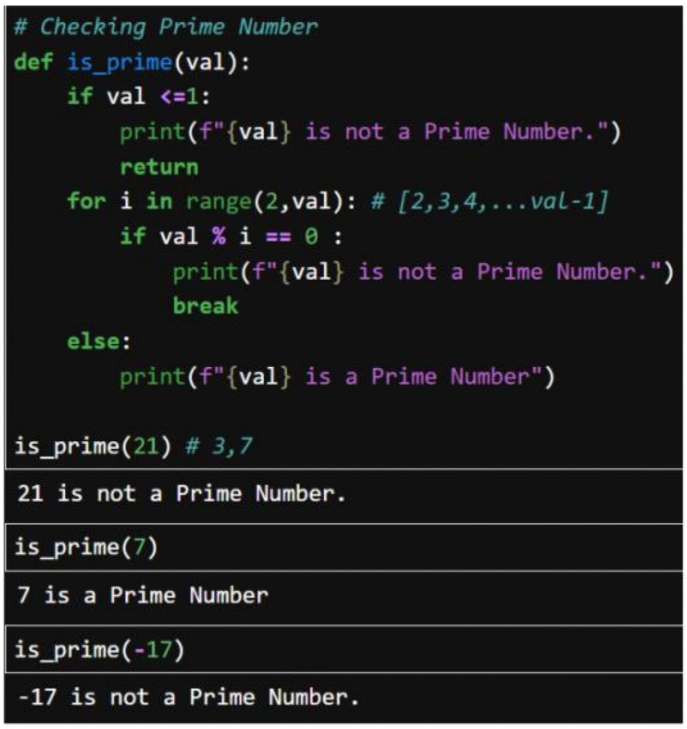
```
2135.0
```

```python
# Arbitary Arguments ❌❌ TypeError: unsupported operand type(s) for +=: 'int' and 'list'
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) # [105.0, 224.0, 354.0, 472.0, 980.0]
print(total_pay)
```



```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) # [105.0, 224.0, 354.0, 472.0, 980.0]
print(total_pay)
```

```
2135.0
```



```python
# Checking Prime Number
def is_prime(val):
    if val <=1:
        print(f"{val} is not a Prime Number.")
        return
    for i in range(2,val): # [2,3,4,...val-1]
        if val % i == 0 :
            print(f"{val} is not a Prime Number.")
            break
    else:
        print(f"{val} is a Prime Number")

is_prime(21) # 3,7
```

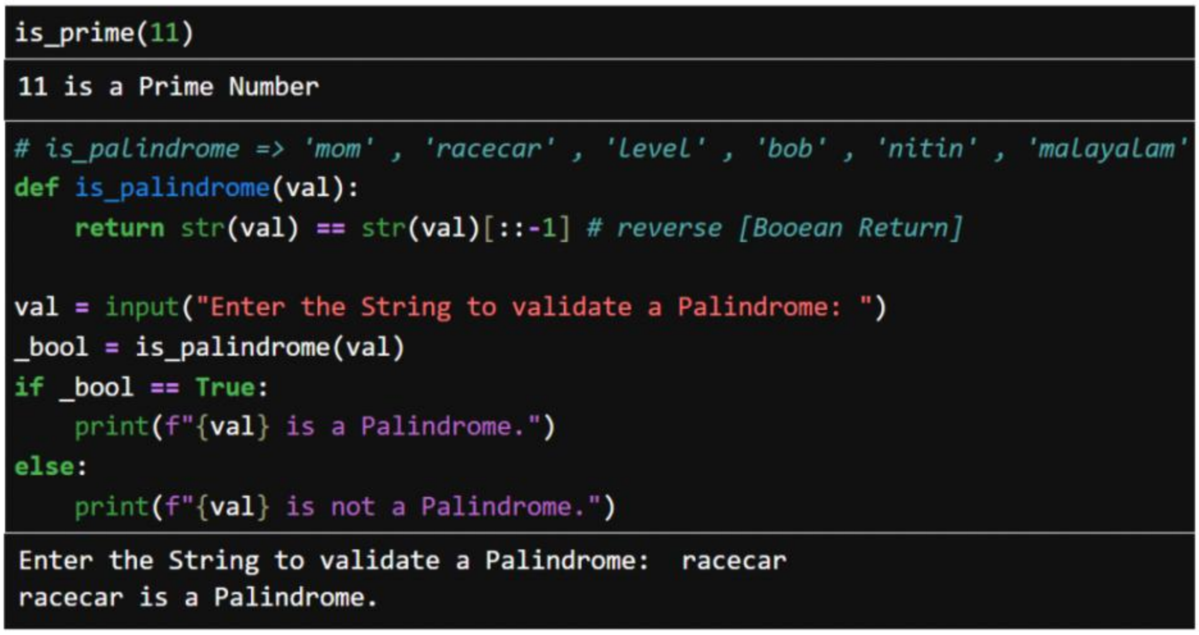
```
21 is not a Prime Number.
```

```python
is_prime(7)
```

```
7 is a Prime Number
```

```python
is_prime(-17)
```

```
-17 is not a Prime Number.
```

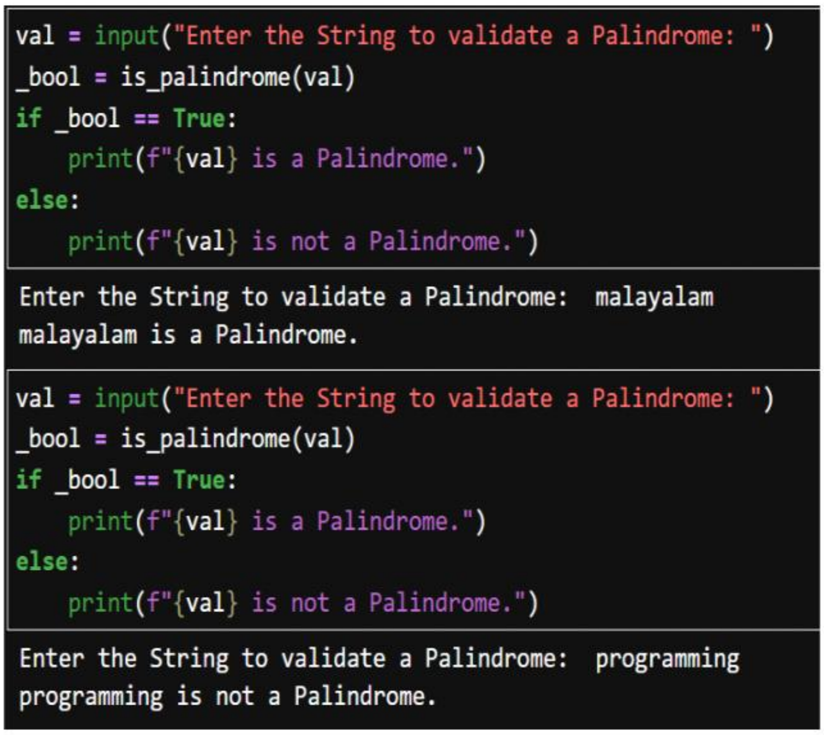


```python
is_prime(11)
```

```
11 is a Prime Number
```

```python
# is_palindrome => 'mom' , 'racecar' , 'level' , 'bob' , 'nitin' , 'malayalam'
def is_palindrome(val):
    return str(val) == str(val)[::-1] # reverse [Booean Return]

val = input("Enter the String to validate a Palindrome: ")
_bool = is_palindrome(val)
if _bool == True:
    print(f"{val} is a Palindrome.")
else:
    print(f"{val} is not a Palindrome.")
```

```
Enter the String to validate a Palindrome:  racecar
racecar is a Palindrome.
```



```python
val = input("Enter the String to validate a Palindrome: ")
_bool = is_palindrome(val)
if _bool == True:
    print(f"{val} is a Palindrome.")
else:
    print(f"{val} is not a Palindrome.")
```

```
Enter the String to validate a Palindrome:  malayalam
malayalam is a Palindrome.
```

```python
val = input("Enter the String to validate a Palindrome: ")
_bool = is_palindrome(val)
if _bool == True:
    print(f"{val} is a Palindrome.")
else:
    print(f"{val} is not a Palindrome.")
```

```
Enter the String to validate a Palindrome:  programming
programming is not a Palindrome.
```