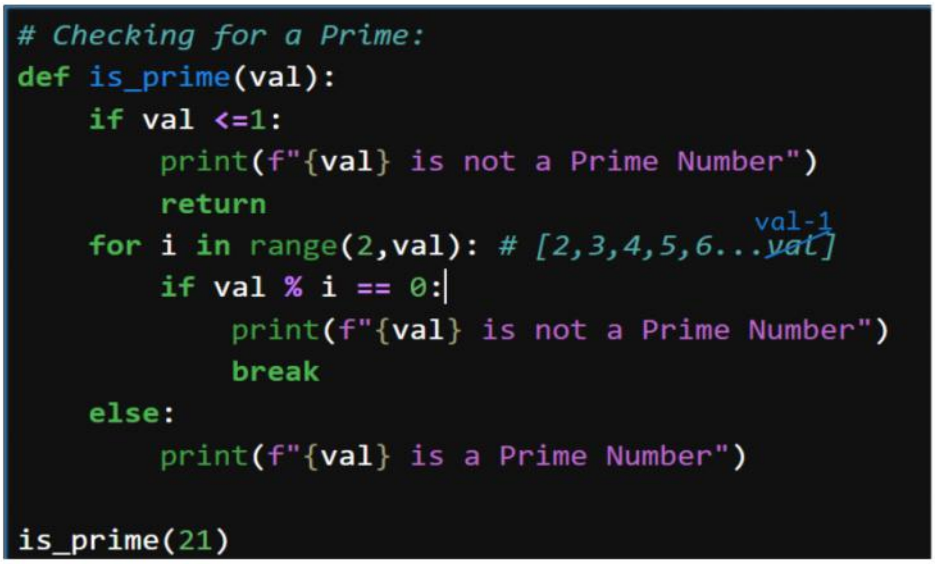
Functions - II

🎯 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.

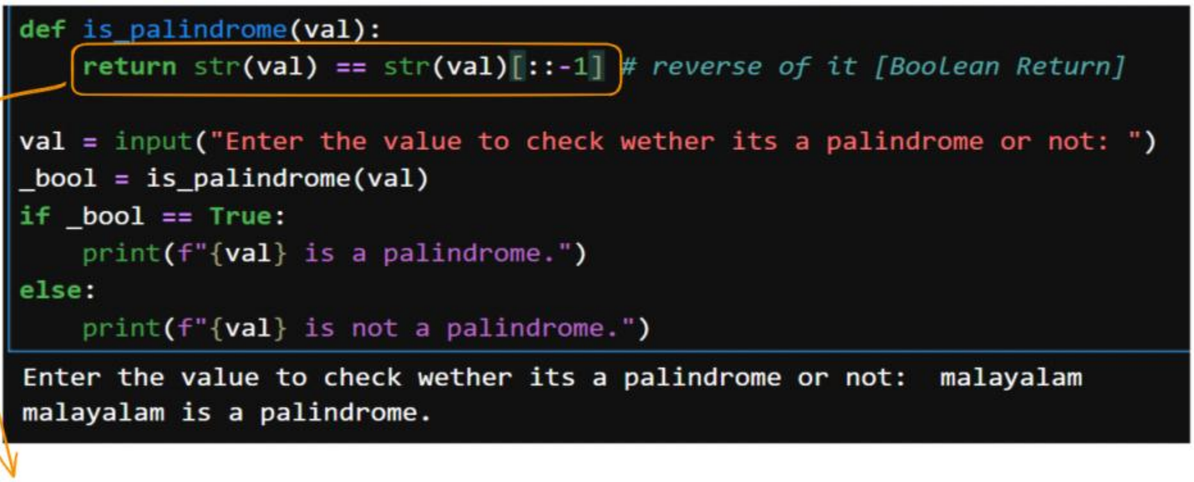


```python
# Checking for a Prime:
def is_prime(val):
    if val <=1:
        print(f"{val} is not a Prime Number")
        return
    for i in range(2,val): # [2,3,4,5,6...val-1 ~~val~~]
        if val % i == 0:
            print(f"{val} is not a Prime Number")
            break
    else:
        print(f"{val} is a Prime Number")

is_prime(21)
```

Memory

val = 27
i = [2,.... val-1]

slicing(start =0, stop = n-1, step = 1)



```python
def is_palindrome(val):
    return str(val) == str(val)[::-1] # reverse of it [Boolean Return]

val = input("Enter the value to check wether its a palindrome or not: ")
_bool = is_palindrome(val)
if _bool == True:
    print(f"{val} is a palindrome.")
else:
    print(f"{val} is not a palindrome.")
```

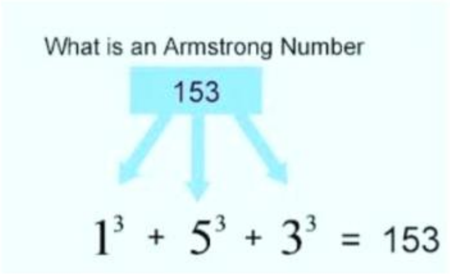
```
Enter the value to check wether its a palindrome or not:  malayalam
malayalam is a palindrome.
```

wow → WOW
0 1 2 → 2 1 0

Phone -> enohP False

Armstrong Number

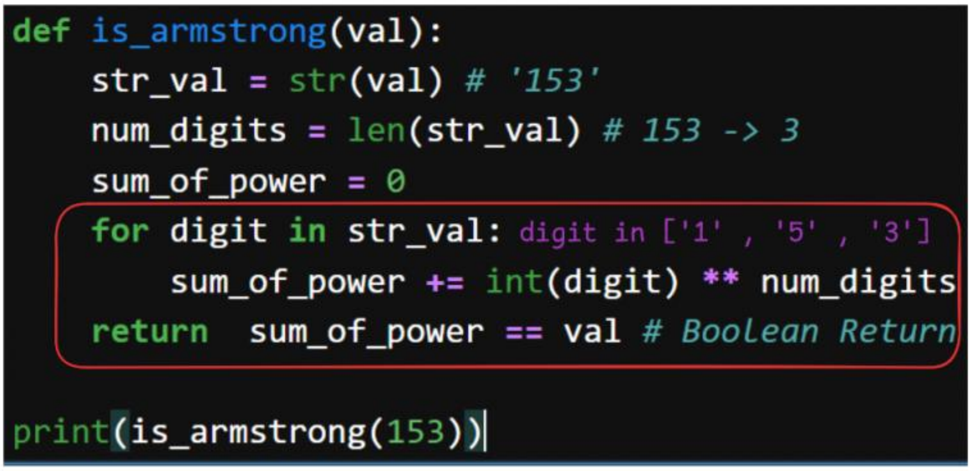
sum of its own digits each raised to the power of the number of digits



$153 = 1^3 + 5^3 + 3^3 = 1 + 125 + 27 = 153$

$371 = 3^3 + 7^3 + 1^3 = 27 + 343 + 1 = 371$

$9474 = 9^4 + 4^4 + 7 + 4^4 = 6561 + 256 + 2401 + 256 = 9474$



```
def is_armstrong(val):
    str_val = str(val) # '153'
    num_digits = len(str_val) # 153 -> 3
    sum_of_power = 0
    for digit in str_val: digit in ['1' , '5' , '3']
        sum_of_power += int(digit) ** num_digits
    return  sum_of_power == val # Boolean Return

print(is_armstrong(153))
```

Memory

val = 153
str_val = '153'
num_digit = 3
sum_of_power = 153
digit = ~~'3'~~ None

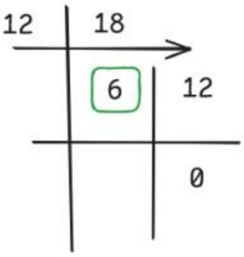
Step1 : Sum_of_power += (1 ** 3) = 0+1 = 1 -> x +=y => x = x + y

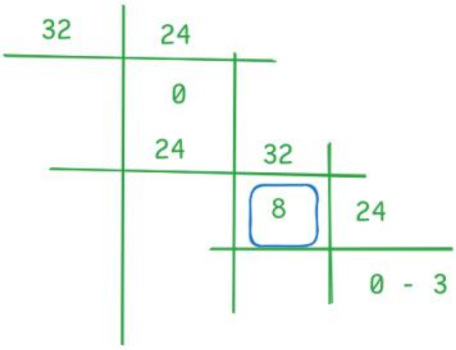
Step2 : Sum_of_power += (5 ** 3) = 1 + 125 = 126

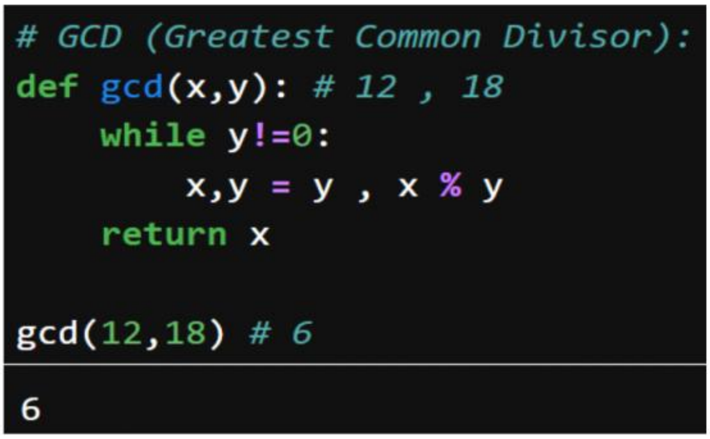
Step3 : Sum_of_power += (3 ** 3) = 126 + 27 = 153

step4 : val == Sum_of_power : True [Console]

GCD

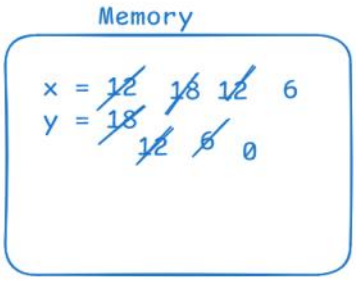


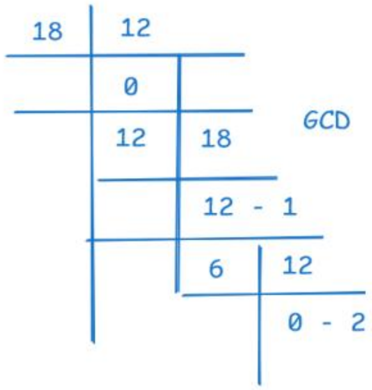


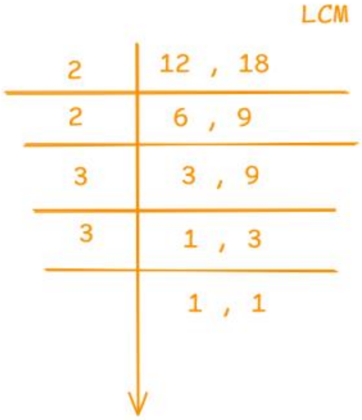


```
# GCD (Greatest Common Divisor):
def gcd(x,y): # 12 , 18
    while y!=0:
        x,y = y , x % y
    return x

gcd(12,18) # 6
```

```
6
```

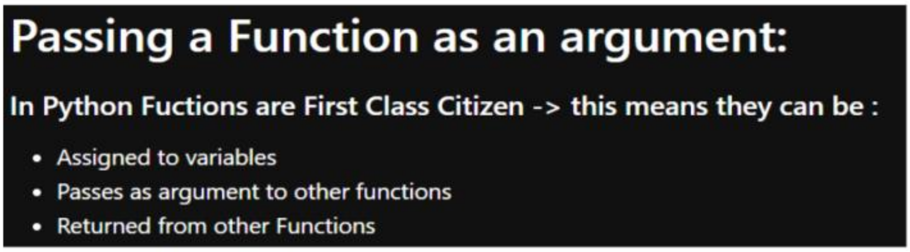






2*2*3*3 = 4 * 9 = 36

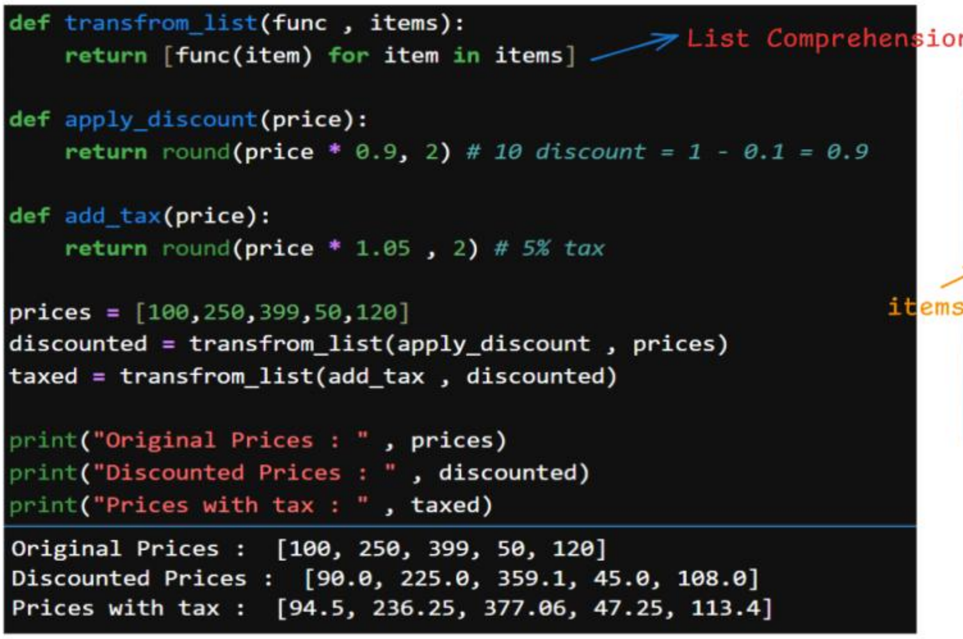
LCM = abs(x*y) // gcd(x,y)



Passing a Function as an argument:

In Python Fuctions are First Class Citizen -> this means they can be :

- Assigned to variables
- Passes as argument to other functions
- Returned from other Functions

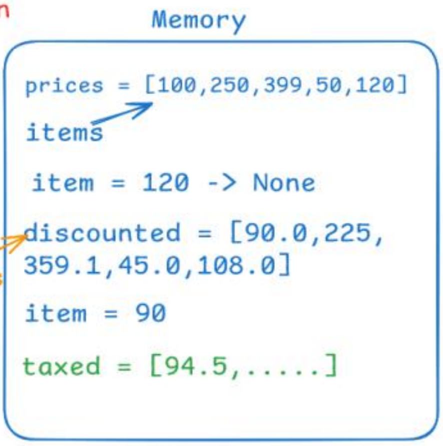


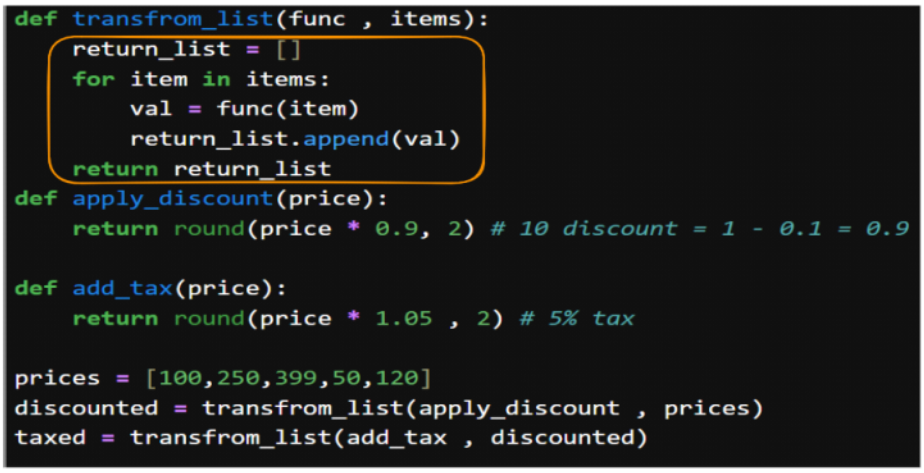
```
def transfrom_list(func , items):
    return [func(item) for item in items]

def apply_discount(price):
    return round(price * 0.9, 2) # 10 discount = 1 - 0.1 = 0.9

def add_tax(price):
    return round(price * 1.05 , 2) # 5% tax

prices = [100,250,399,50,120]
discounted = transfrom_list(apply_discount , prices)
taxed = transfrom_list(add_tax , discounted)

print("Original Prices : " , prices)
print("Discounted Prices : " , discounted)
print("Prices with tax : " , taxed)
```

```
Original Prices :  [100, 250, 399, 50, 120]
Discounted Prices :  [90.0, 225.0, 359.1, 45.0, 108.0]
Prices with tax :  [94.5, 236.25, 377.06, 47.25, 113.4]
```

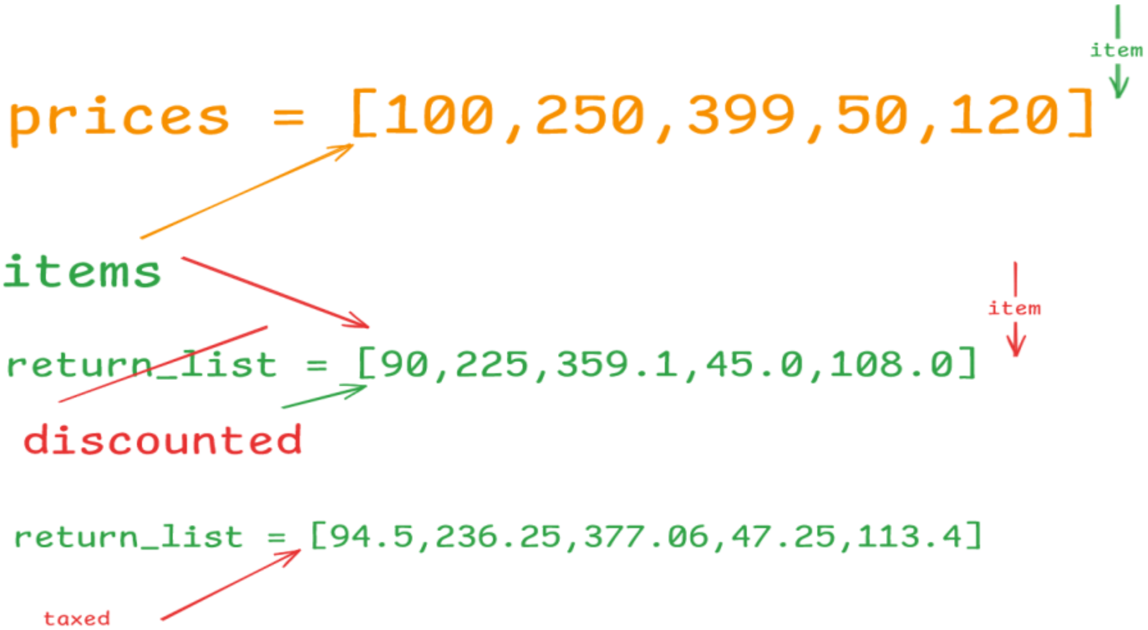
List Comprehension

items

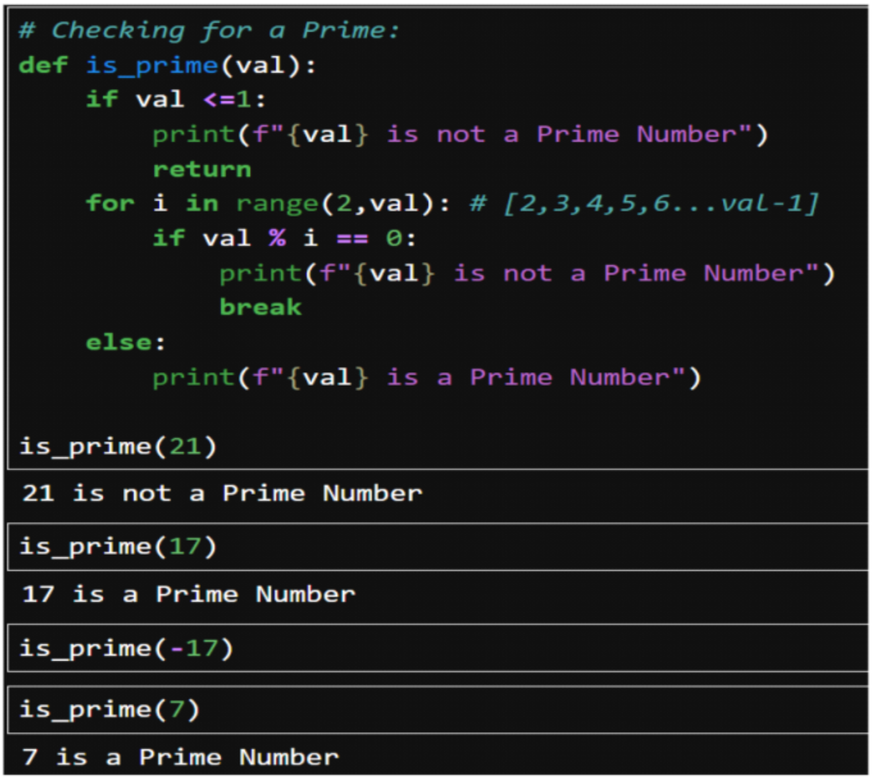




```python
def transfrom_list(func , items):
    return_list = []
    for item in items:
        val = func(item)
        return_list.append(val)
    return return_list
def apply_discount(price):
    return round(price * 0.9, 2) # 10 discount = 1 - 0.1 = 0.9

def add_tax(price):
    return round(price * 1.05 , 2) # 5% tax

prices = [100,250,399,50,120]
discounted = transfrom_list(apply_discount , prices)
taxed = transfrom_list(add_tax , discounted)
```



item
prices = [100,250,399,50,120]
items
item
return_list = [90,225,359.1,45.0,108.0]
discounted
return_list = [94.5,236.25,377.06,47.25,113.4]
taxed



```python
# Checking for a Prime:
def is_prime(val):
    if val <=1:
        print(f"{val} is not a Prime Number")
        return
    for i in range(2,val): # [2,3,4,5,6...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)
```

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

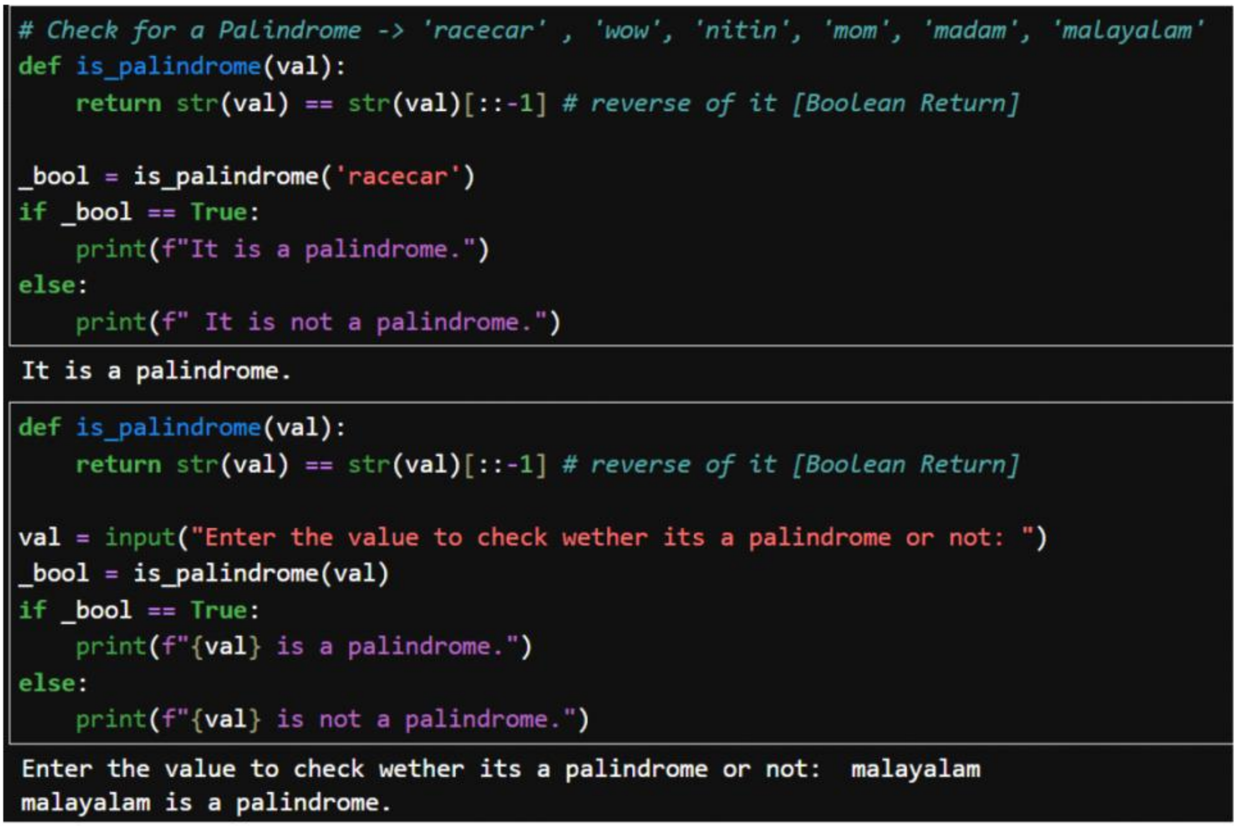
```python
is_prime(17)
```

```
17 is a Prime Number
```

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

```python
is_prime(7)
```

```
7 is a Prime Number
```

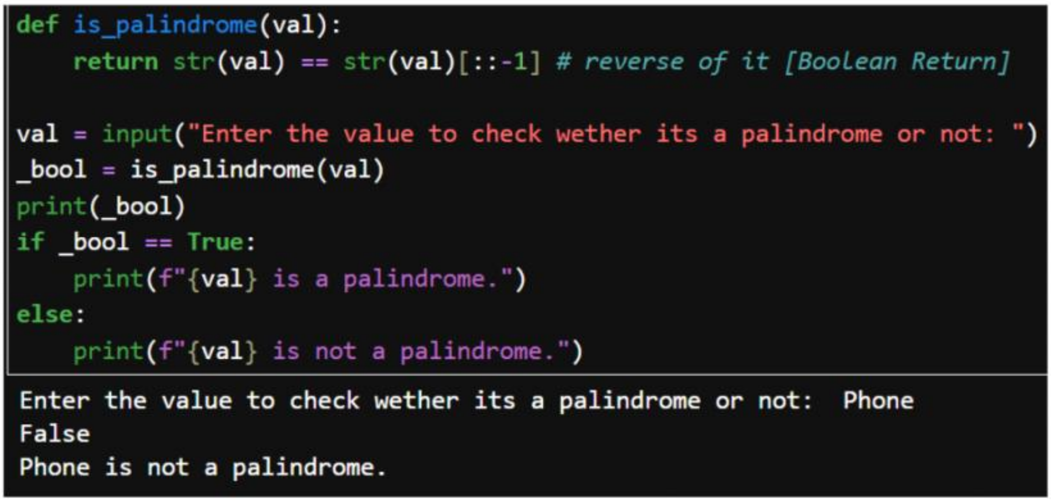


```
# Check for a Palindrome -> 'racecar' , 'wow', 'nitin', 'mom', 'madam', 'malayalam'
def is_palindrome(val):
    return str(val) == str(val)[::-1] # reverse of it [Boolean Return]

_bool = is_palindrome('racecar')
if _bool == True:
    print(f"It is a palindrome.")
else:
    print(f" It is not a palindrome.")
```

```
It is a palindrome.
```

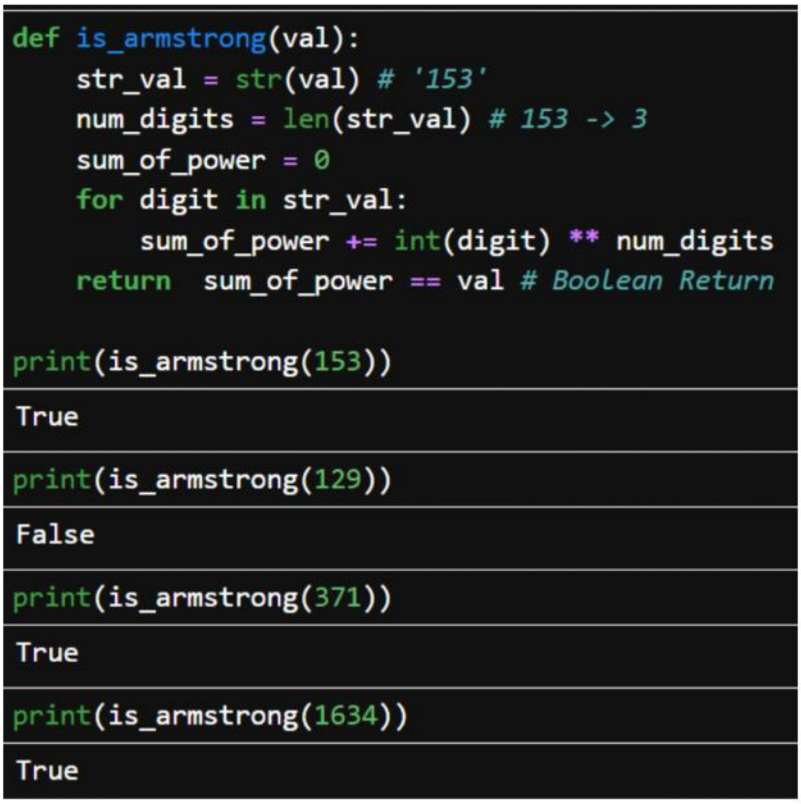
```
def is_palindrome(val):
    return str(val) == str(val)[::-1] # reverse of it [Boolean Return]

val = input("Enter the value to check wether its a palindrome or not: ")
_bool = is_palindrome(val)
if _bool == True:
    print(f"{val} is a palindrome.")
else:
    print(f"{val} is not a palindrome.")
```

```
Enter the value to check wether its a palindrome or not:  malayalam
malayalam is a palindrome.
```



```
def is_palindrome(val):
    return str(val) == str(val)[::-1] # reverse of it [Boolean Return]

val = input("Enter the value to check wether its a palindrome or not: ")
_bool = is_palindrome(val)
print(_bool)
if _bool == True:
    print(f"{val} is a palindrome.")
else:
    print(f"{val} is not a palindrome.")
```

```
Enter the value to check wether its a palindrome or not:  Phone
False
Phone is not a palindrome.
```



```
def is_armstrong(val):
    str_val = str(val) # '153'
    num_digits = len(str_val) # 153 -> 3
    sum_of_power = 0
    for digit in str_val:
        sum_of_power += int(digit) ** num_digits
    return  sum_of_power == val # Boolean Return

print(is_armstrong(153))
```

```
True
```

```
print(is_armstrong(129))
```

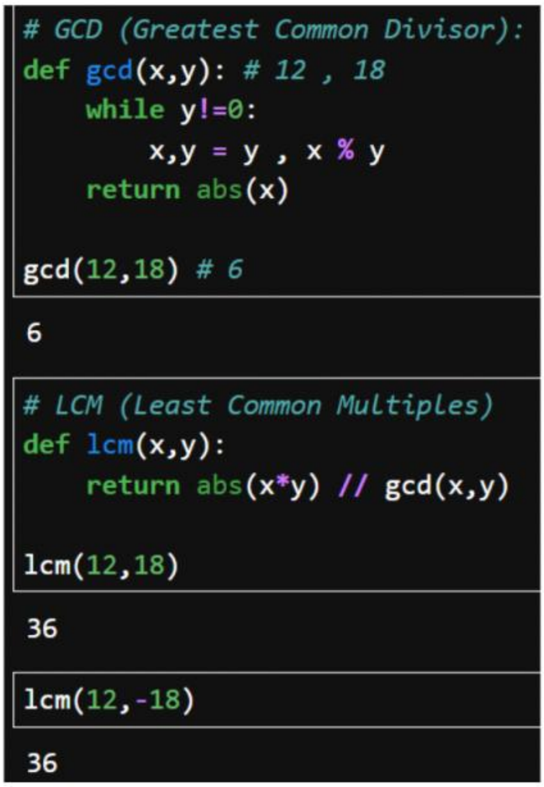
```
False
```

```
print(is_armstrong(371))
```

```
True
```

```
print(is_armstrong(1634))
```

```
True
```



```
# GCD (Greatest Common Divisor):
def gcd(x,y): # 12 , 18
    while y!=0:
        x,y = y , x % y
    return abs(x)

gcd(12,18) # 6
```

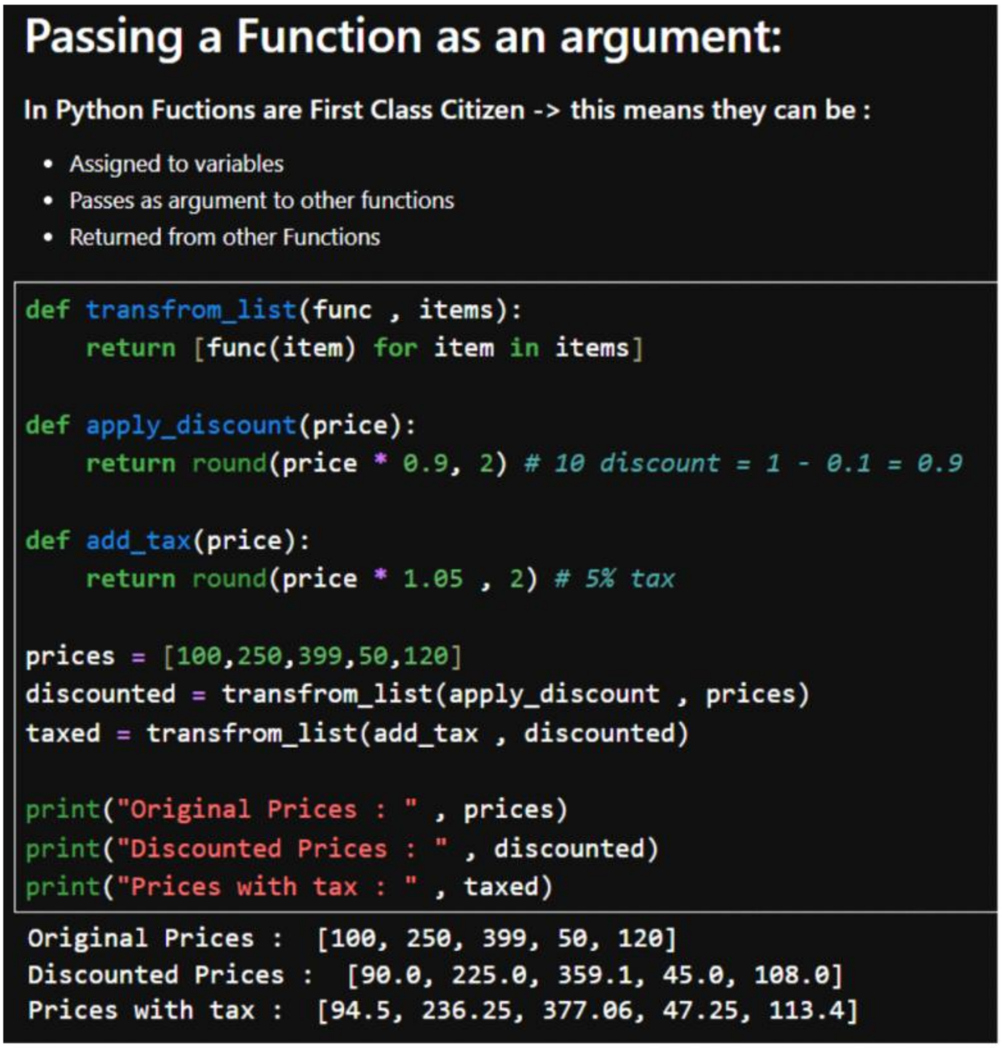
```
6
```

```
# LCM (Least Common Multiples)
def lcm(x,y):
    return abs(x*y) // gcd(x,y)

lcm(12,18)
```

```
36
```

```
lcm(12,-18)
```

```
36
```



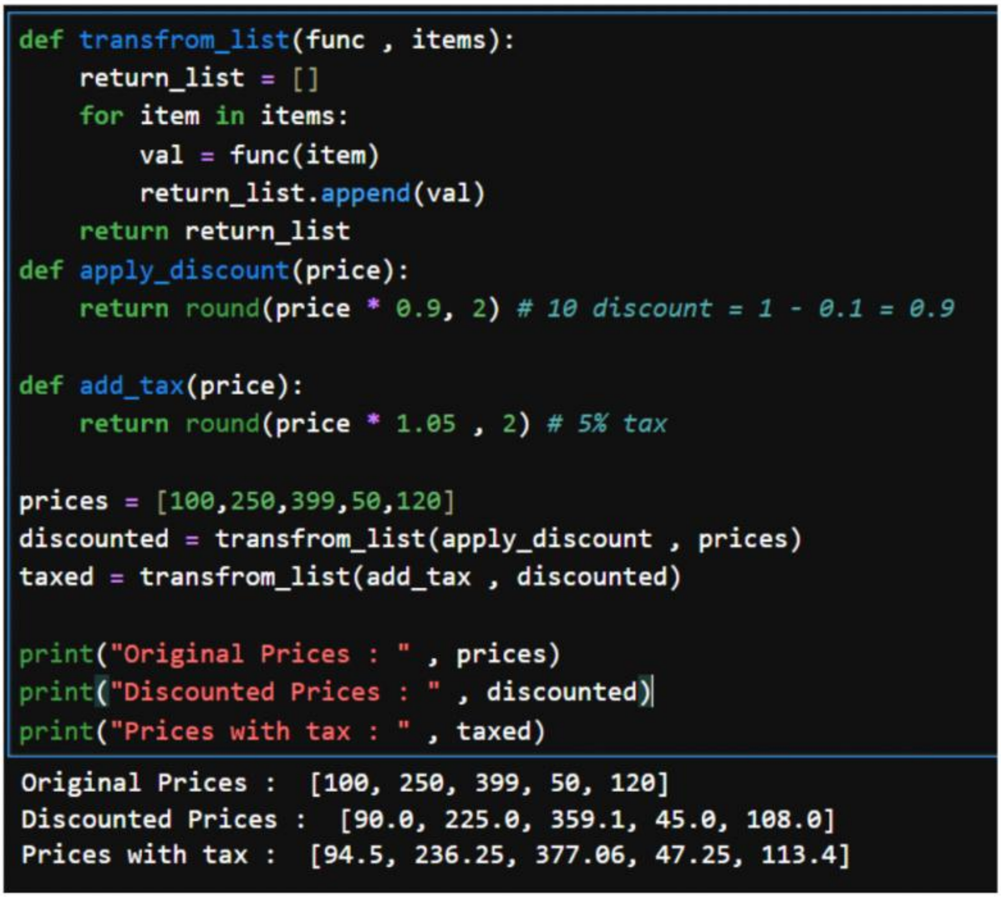
Passing a Function as an argument:

In Python Fuctions are First Class Citizen -> this means they can be :

- Assigned to variables
- Passes as argument to other functions
- Returned from other Functions

```
def transfrom_list(func , items):
    return [func(item) for item in items]

def apply_discount(price):
    return round(price * 0.9, 2) # 10 discount = 1 - 0.1 = 0.9

def add_tax(price):
    return round(price * 1.05 , 2) # 5% tax

prices = [100,250,399,50,120]
discounted = transfrom_list(apply_discount , prices)
taxed = transfrom_list(add_tax , discounted)

print("Original Prices : " , prices)
print("Discounted Prices : " , discounted)
print("Prices with tax : " , taxed)
```

```
Original Prices :  [100, 250, 399, 50, 120]
Discounted Prices :  [90.0, 225.0, 359.1, 45.0, 108.0]
Prices with tax :  [94.5, 236.25, 377.06, 47.25, 113.4]
```



```
def transfrom_list(func , items):
    return_list = []
    for item in items:
        val = func(item)
        return_list.append(val)
    return return_list
def apply_discount(price):
    return round(price * 0.9, 2) # 10 discount = 1 - 0.1 = 0.9

def add_tax(price):
    return round(price * 1.05 , 2) # 5% tax

prices = [100,250,399,50,120]
discounted = transfrom_list(apply_discount , prices)
taxed = transfrom_list(add_tax , discounted)

print("Original Prices : " , prices)
print("Discounted Prices : " , discounted)
print("Prices with tax : " , taxed)
```

```
Original Prices :  [100, 250, 399, 50, 120]
Discounted Prices :  [90.0, 225.0, 359.1, 45.0, 108.0]
Prices with tax :  [94.5, 236.25, 377.06, 47.25, 113.4]
```