```
def evenOrOdd_params_noreturn(a):
    if a % 2 == 0:
        print("Even")
    else:
        print("Odd")


def evenOrOdd_params_return(a):
    if a % 2 == 0:
        return "Even"
    else:
        return "Odd"


def evenOrOdd_no_params_noreturn():
    a = 10
    if a % 2 == 0:
        print("Even")
    else:
        print("Odd")


def evenOrOdd_no_params_return():
    a = 10
    if a % 2 == 0:
        return "Even"
    else:
        return "Odd"


evenOrOdd_no_params_noreturn()
evenOrOdd_params_noreturn(10)
print(evenOrOdd_params_return(10))
print(evenOrOdd_no_params_return())


# Types of parameters
# function decalartion
def greetings(greet, to="Vinay", time="good morning"):
    return greet + " " + to + " " + time


# function call
print(greetings("Hello", time="good ni8"))


def grocery_store(customer_name, items_with_prices, *items,
delivery_type="in-store", **items_with_qty):
    print("customer name is " + customer_name)
    print("Items are" + str(items))
    print(type(items_with_qty))
    print("Items with_qty are" + str(items_with_qty))
    print("Delivery mode is " + delivery_type)
    print("Items with price are " + str(items_with_prices))
```

```
items_with_price = {"egg": 36, "milk": "45"}
items_with_qty = {"eggs": 12, "milk": "1 ltr"}
item_list = ["Toothpaste", "bread", "eggs", "maggi"]
grocery_store("jyothi", items_with_price, *item_list,
**items_with_qty, delivery_type="home")


# lambda function
def add(x, y):
    return x + y


func = lambda x, y: x + y
print(func(5, 7))


def apply_ops(*args, ops):
    return ops(args)


print(apply_ops(2, 3, 7, 9, ops=sum))
print(apply_ops(2, 3, 7, 9, ops=len))
print(apply_ops(2, 3, 7, 9, ops=max))

num_list = [[2, 8, 11], [9, 5, 7, 12], [8, 9, 10, 11], [1, 13, 17],
[2, 5, 16]]
print(max(num_list, key=sum))
print(max(num_list, key=lambda x: x[0]))

# map function
num_list = [2, 8, 11, 4, 5, 7, 12]
list2 = list(map(lambda x: x * x, num_list))
print(list2)
print(num_list)


# reduce and filter

# closures
# decorators / Generators [Advance Python]
def outer_func():
    def inner_func():
        return ("Inside inner func")

    return inner_func


var = outer_func()
print(var())


def to_power(x):
    def calculate_power(n):
        return n ** x

    return calculate_power
```

```
find_values = lambda x, n: x ** n
print(find_values(5, 3))

cube = to_power(3)
print(cube(5))
square = to_power(2)
print(square(10))
quad = to_power(4)
print(quad(15))
```