```python
#task 3 storing in list using function
users=[]
def full_name(first_name,last_name):
    name=f"{first_name} {last_name}"
    users.append(name)
    return users

active=True
while active:
    print("please enter your first name and last name")
    print("Type quit to exit at any time")
    f_name=input("First name= ")
    if f_name.lower() == 'quit':
        break
    l_name=input("Last anme= ")
    if l_name.lower() == 'quit':
        break
    people=full_name(f_name,l_name)

print(people)
```

```
please enter your first name and last name
Type quit to exit at any time
First name= ankit
Last anme= nayan
please enter your first name and last name
Type quit to exit at any time
First name= ayush
Last anme= nayan
please enter your first name and last name
Type quit to exit at any time
First name= quit
['ankit nayan', 'ayush nayan']
```

```python
for name in people:
    print(name)
```

```
ankit nayan
ayush nayan
```

```python
for name in people:
    print(name,end=' ')
```

```
ankit nayan ayush nayan
```

```python
# passing arbitrary no of arguments
def function_name(*parameters)

def make_pizza(size,*toppings):  # you can take as many no of argument
#when it is one star it is passed as list
    print(f"Making {size} = inches pizza with following toppings")
    for topping in toppings:
```

```
        print(f"{topping.title()}")
make_pizza(10, 'extra cheese')
make_pizza(12, 'extra cheese', 'capsicum', 'jalapeno')
```

```
Making 10 = inches pizza with following toppings
Extra Cheese
Making 12 = inches pizza with following toppings
Extra Cheese
Capsicum
Jalapeno
```

```
# arbitrary key word arguments
def make_pizza(size,**toppings):   # #when it is two star it is passed
as dictionary
    print(f"Making {size} = inches pizza with following toppings")
    for topping in toppings:
        print(f"{topping.title()}")
make_pizza(10, first_topping='extra cheese')
make_pizza(12, first_topping='extra cheese',
second_topping='capsicum',third_topping= 'jalapeno')
```

```
Making 10 = inches pizza with following toppings
First_Topping
Making 12 = inches pizza with following toppings
First_Topping
Second_Topping
Third_Topping
```

it is just printing keys

not the value in above code

```
# arbitrary key word arguments
def make_pizza(size,**toppings):
    print(f"Making {size} = inches pizza with following toppings")
    for v in toppings.values():
        print(f"Toppings = {v.title()}")
make_pizza(10, first_topping='extra cheese')
make_pizza(12, first_topping='extra cheese',
second_topping='capsicum',third_topping= 'jalapeno')
```

```
Making 10 = inches pizza with following toppings
Toppings = Extra Cheese
Making 12 = inches pizza with following toppings
Toppings = Extra Cheese
Toppings = Capsicum
Toppings = Jalapeno
```

```
def car_info(manufacturer,model_name,**other_information): #here it is
two start
    car={
        'manufacturer':manufacturer,
        'model_name':model_name,
        'misc_info':other_information
    }
    return car

#task
# call this car info function with the value
# manufacturer = tesla
# model = s
#color = white
# type = sedan
# vehicle_power = battery

# print the information on the console
car=car_info('tesla','s',color='white',type='sedan',vehivle_power='bat
tery')

print(f"Manufacturer - {car['manufacturer'].title()}")
print(f"Model - {car['model_name'].title()}")
```

```
Manufacturer - Tesla
Model - S
```

```
for k,v in car['misc_info'].items():
    print(f"{k.title()} - {v.title()}")#
```

```
Color - White
Type - Sedan
Vehivle_Power - Battery
```

```
# write code in this format
car={
    'manufacturer':'tesla',
    'model_name':'s',
    'misc_info':{
        'color'='white',
        'type'='sedan',
        'vehicle_power'='battery'
    }
}
```

Modules

modules help you to use your functions in different files and thus separate the implementation logic and execution logic

#module is file ending in .py file

task 1

```
# kitchen.py
def make_pizza(size, *toppings):
    print(f"Making {size} inches pizza with following toppings")
    for topping in toppings:
        print(f"{topping.title()}")

#restaurent.py
import kitchen as k
k.make_pizza(12,'capsicum','olives','jalapeno')
```

```
---------------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
<ipython-input-35-dce4d7bd1c14> in <module>
      1 #restaurent.py
----> 2 import kitchen as k
      3 k.make_pizza(12,'capsicum','olives','jalapeno')

ModuleNotFoundError: No module named 'kitchen'
```

task 2

```
# kitchen.py
def make_pizza(size, *toppings):
    print(f"Making {size} inches pizza with following toppings")
    for topping in toppings:
        print(f"{topping.title()}")
def make_chocolate_milk_shakes(size,*ingredients):
    print(f"Here is your {size} chocolate size")
    for ingredient in ingredients:
        print(f"{ingredients.title()}")
def day_info():
    print("today is wednesday")
def month_info():
    print("may is the month name")

# restaurent.py
import kitchen as k
k.make_pizza(12,'capsicum','olives','jalapeno')
print()
k.make_chocolate_milk_shake('large','chocolate ice
cream','milk','ice')
print()
k.day_info()

# Method2 another way without using alias # when u have long module
name u can use alias
import kitchen
```

```
kitchen.make_pizza(12,'capsicum','olives','jalapeno')
print()
kitchen.make_chocolate_milk_shake('large','chocolate ice
cream','milk','ice')
print()
kitchen.day_info()
```

task 3

```
# kitchen.py
def make_pizza(size, *toppings):
    print(f"Making {size} inches pizza with following toppings")
    for topping in toppings:
        print(f"{topping.title()}")
def make_chocolate_milk_shakes(size,*ingredients):
    print(f"Here is your {size} chocolate size")
    for ingredient in ingredients:
        print(f"{ingredients.title()}")
def day_info():
    print("today is wednesday")
def month_info():
    print("may is the month name")

# restaurent1 # method 1
import kitchen as k
k.make_pizza(12,'capsicum','olives','jalapeno')
print()
k.make_chocolate_milk_shake('large','chocolate ice
cream','milk','ice')
print()
k.day_info()

# restaurent 2   # method 2
from kitchen import *
make_pizza(10,'capsicum')
make_chocolate_milk_shake('small','chocolate','milk','ice')
day_info()
month_info()

# restaurent 3   # method 3
from kitchen import make_pizza,make_chocolate_milk_shake
make_pizza(10,'capsicum')
make_chocolate_milk_shake('small','chocolate','milk','ice')
day_info()
month_info()
```

Diffrent method

import module as alias

import module

from module import function

from module import *

from module import function as fn

```
from kitchen import make_chocolate_milk_shake as ch
ch('medium','ice cream','milk')
```

TASK 5

implement in pycharm

```
# pizza.py
def make_pizza(size, *toppings):
    print(f"Making {size} inches pizza with following toppings")
    for topping in toppings:
        print(f"{topping.title()}")

# chocolate_milk_shakes.py
def make_chocolate_milk_shakes(size,*ingredients):
    print(f"Here is your {size} chocolate size")
    for ingredient in ingredients:
        print(f"{ingredients.title()}")

# dayinfo.py
def day_info():
    print("today is wednesday")

# monthinfo.py
def month_info():
    print("may is the month name")
```

Method 1

from module_name import function_name

```
from pizza import make_pizza
from chocolate_milk_shake.py import make_chocolate_milk_shake
from dayinfo import day_info
from monthinfo import month_info

make_pizza(10,'capsicum')
make_chocolate_milk_shake('small','chocolate','milk','ice')
day_info()
month_info()
```

Method 2

import module_name

```
import pizza
import chocolate_milk_shakes
import dayinfo
import monthinfo

pizza.make_pizza(12,'capsicum','olives','jalapeno')
print()
chocolate_milk_shakes.make_chocolate_milk_shake('large','chocolate ice
cream','milk','ice')
print()
day_info.day_info()
```

Method 3

from module_name import function_name as fn

```
from pizza import make_pizza as mp
from chocolate_milk_shake.py import make_chocolate_milk_shake as mcms
from dayinfo import day_info as di
from monthinfo import month_info as mi

pizza.mp(12,'capsicum','olives','jalapeno')
print()
chocolate_milk_shakes.mcms('large','chocolate ice cream','milk','ice')
print()
day_info.di()
```

Method 4

import module_name as mn

```
import pizza as p
import chocolate_milk_shakes as cms
import dayinfo as di
import monthinfo as mi

p.make_pizza(12,'capsicum','olives','jalapeno')
print()
cms.make_chocolate_milk_shake('large','chocolate ice
cream','milk','ice')
print()
di.day_info()
```

Method 5

from module_name import *

```
from pizza import *
from chocolate_milk_shakes import *
from dayinfo import *
from monthinfo import *

make_pizza(10,'capsicum')
make_chocolate_milk_shake('small','chocolate','milk','ice')
day_info()
month_info()
```

```
---------------------------------------------------------------------------
-----
ModuleNotFoundError                       Traceback (most recent call
last)
<ipython-input-37-bb4ec3c85dd3> in <module>
----> 1 from pizza import *
      2 from pizza import *
      3 from pizza import *

ModuleNotFoundError: No module named 'pizza'
```

Lambda function

```
def add_number(num):
    value=num+20
    return value
```

```
result = add_number(20)
print(result)
```

```
40
```

```
result=  lambda     x    :   x+20
#Syntax:= lambda argument         function

x=lambda a,b:a*b
print(x(10,20))
```

```
200
```

```
# More exapmles

# list of even numbers
# function as a parameter
numbers = [20,23,1,4,99,21,32,44,67,98,80]
even_numbers=list(filter(lambda x: (x%2==0),numbers))
#syntax:-filter(lambda  function  , numbers)
print(even_numbers)
```

```
[20, 4, 32, 44, 98, 80]
```

```
numbers = [20,23,1,4,99,21,32,44,67,98,80]
even_numbers=list(filter(lambda x: (x%2==0),numbers))
for even_number in even_numbers:
    print(even_number)
```

```
20
4
32
44
98
80
```

```
# print full name
full_name=lambda first,last: f'Full name: {first.title()}
{last.title()}'
print(full_name)
```

```
<function <lambda> at 0x000001D27153F8B0>
```

```
# Task
# from the list of numbers =[20,23,1,4,99,21,32,44,67,98,80] print all
those functions greater than 10

numbers =[20,23,1,4,99,21,32,44,67,98,80]
greater_than_10=list(filter(lambda x:x>10,numbers))
print(greater_than_10)
```

```
[20, 23, 99, 21, 32, 44, 67, 98, 80]
```

```
greater_than_10=list(map(lambda x:x>10,
[20,23,1,4,99,21,32,44,67,98,80]))
greater_than_10
```

```
True
True
False
False
True
True
True
True
True
True
True
```

```
[None, None, None, None, None, None, None, None, None, None, None]
```

```
# map function
number = [1,5,32,45,88,20]
new_numbers=list(map(lambda x: x*10,numbers))
print(new_numbers)
```

```
[200, 230, 10, 40, 990, 210, 320, 440, 670, 980, 800]
```

```
# square of numbers
list_numbers=[1,5,32,45,88,20]
square_of_numbers=list(map(lambda x:pow(x,2),list_numbers))
square_of_numbers.sort()
print(square_of_numbers)
```

```
[1, 25, 400, 1024, 2025, 7744]
```

```
# Global Variable
full_name = '' # when you write at the top then also it becomes global
def print_full_name(first_name,last_name):
    full_name=f"{first_name.title()}{last_name.title()}"
    print(full_name)
def day_info():
    print("Today is Wednesday")
    print(full_name)

full_name = ''
def print_full_name(first_name,last_name):
    global full_name  # now full_name is global to all the function
    full_name=f"{first_name.title()}{last_name.title()}"
    print(full_name)
def day_info():
    print("Today is Wednesday")
    print(full_name)
```

```
print_full_name('james','bond')
day_info()
```

```
JamesBond
Today is Wednesday
JamesBond
```

```
# question - full_name='donald_duck'
full_name = ''
def print_full_name(first_name,last_name):
    global full_name  # now full_name is global to all the function,
in day info and month info as welll
    full_name=f"{first_name.title()}{last_name.title()}"
    print(full_name)

def day_info():
    print("Today is Wednesday")
    full_name-'Donald Duck'  # this is getting updated locally
    print(full_name)

def month_info():
    print(full_name)
print_full_name('james','bond')
day_info()
month_info()
```

```
JamesBond
Today is Wednesday

---------------------------------------------------------------------------
-----
TypeError                                 Traceback (most recent call
last)
<ipython-input-59-3f7e7cd5e4ad> in <module>
     14     print(full_name)
     15 print_full_name('james','bond')
---> 16 day_info()
     17 month_info()

<ipython-input-59-3f7e7cd5e4ad> in day_info()
      8 def day_info():
      9     print("Today is Wednesday")
---> 10     full_name-'Donald Duck'
     11     print(full_name)
     12

TypeError: unsupported operand type(s) for -: 'str' and 'str'
```