1. Delete odd elements

```
list = [11, 22, 33, 44, 55]

for i in list:
    if ( i % 2 != 0 ) :
        list.remove(i)

print (list)
```

2. Divisble by 3

```
for i in range (1,20):
    if i % 3==0:
        print (i)
```

3. Divisible by 4

```
for i in range (1,51):
    if i % 4==0:
        print (i)
```

4. Factorial

```
num = int(input("Enter the Number:"))
n1, n2 = 0, 1
print("Fibonacci Series:", n1, n2, end=" ")
for i in range(2, num):
    n3 = n1 + n2
    n1 = n2
    n2 = n3
    print(n3, end=" ")
print()
```

5. Factorial using while loop

```
num = int(input("enter a number: "))
fac = 1
i = 1
while i <= num:
    fac = fac * i
    i = i + 1
print("factorial of ", num, " is ", fac)
```

6. File handling

```
import collections                                    Analyzing...
import pprint
with open("MyData", 'r') as data:
    count_data = collections.Counter(data.read().upper())
    count_value = pprint.pformat(count_data)
print(count_value)
```

7. Hybrid Inheritance

```
class School:                                         ⚠ 5
    def func1(self):
        print("This function is in school.")

class Student1(School):
    def func2(self):
        print("This function is in student 1. ")

class Student2(School):
    def func3(self):
        print("This function is in student 2.")

class Student3(Student1, School):
    def func4(self):
        print("This function is in student 3.")

object = Student3()
object.func1()
object.func2()
```

8. Import module package

```
# import all names from the standard module math

from math import *
print("The value of pi is", pi)
```

9. Insert at end

```
lst=[2,4,6]
lst.append(7)
lst.append(8)
print("The appended list is:",lst)
```

10. Check number is positive , negative or zero

```
num = float(input("Input a number: "))
if num > 0:
    print("It is positive number")
elif num == 0:
    print("It is Zero")
else:
    print("It is a negative number")
```

11. Menu driven calculator

```
def add(a, b):
    sum = a + b
    print(a, "+", b, "=", sum)
def subtract(a, b):
    difference = a - b
    print(a, "-", b, "=", difference)
def multiply(a, b):
    product = a * b
    print(a, "x", b, "=", product)
def divide(a, b):
    division = a / b
    print(a, "/", b, "=", division)
print("++++++++++++++++++++++++++++++++")
print("A SIMPLE CALCULATOR")
print("++++++++++++++++++++++++++++++++")
while True:
    print("MENU")
    print("1. Addition")
    print("2. Subtraction")
    print("3. Multiplication")
    print("4. Division")
    print("5. Exit")
    choice = int(input("\nEnter the Choice: "))
    print("++++++++++++++++++++")
    if choice == 1:
        print("\nADDITION")
        a = int(input("First Number: "))
        b = int(input("Second Number: "))
        add(a, b)
    elif choice == 2:
        print("\nSUBTRACTION")
        a = int(input("First Number: "))
        b = int(input("Second Number: "))
        subtract(a, b)
    elif choice == 3:
        print("\nMULTIPLICATION")
        a = int(input("First Number: "))
        b = int(input("Second Number: "))
        multiply(a, b)
    elif choice == 4:
        print("\nDIVISION")
        a = int(input("First Number: "))
        b = int(input("Second Number: "))
        divide(a, b)
```

```
    elif choice == 5:
        break

    else:
        print("Please Provide a valid Input!")
        print("++++++++++++++++++++++++++++")
```

12. Multi-level inheritance

```
class GrandFather:
    def ownHouse(self):
        print("Grandpa House")

class Father(GrandFather):
    def ownBike(self):
        print("Father's Bike")

class Son(Father):
    def ownBook(self):
        print("Son Have a Book")

object = Son()
object.ownHouse()
object.ownBike()
object.ownBook()
```

13. Multiple inheritance

```python
class Class1:
    def m(self):
        print("In Class1")

class Class2(Class1):
    def m(self):
        print("In Class2")

class Class3(Class1):
    def m(self):
        print("In Class3")

class Class4(Class2, Class3):
    pass

obj = Class4()
obj.m()
```

14. Multi-thread

```python
import threading
import time

def useless_function(seconds):
    print(f'Waiting for {seconds} second(s)', end = "\n")
    time.sleep(seconds)
    print(f'Done Waiting {seconds}  second(s)')
start = time.perf_counter()
t = threading.Thread(target=useless_function, args=[1])
t.start()
print(f'Active Threads: {threading.active_count()}')
t.join()
end = time.perf_counter()
print(f'Finished in {round(end-start, 2)} second(s)')
```

15. Numpymatplot

```
import matplotlib.pyplot as plt

slices = [7, 2, 2, 13]
activities=['sleeping', 'eating', 'working', 'playing']
cols = ['c', 'm', 'r', 'b']

plt.pie(slices,
        labels=activities,
        colors=cols,
        startangle=90,
        shadow=True,
        explode=(0, 0, 0, 0),
        autopct='%1.1f%%')
plt.title('Pie Plot')
plt.show()
```

16. Polymorphism using class

```
class India():
    def capital(self):
        print("New Delhi is the capital of India.")
    def language(self):
        print("Hindi is the most widely spoken language of
    def type(self):
        print("India is a developing country.")
class USA():
    def capital(self):
        print("Washington, D.C. is the capital of USA.")
    def language(self):
        print("English is the primary language of USA.")
    def type(self):
        print("USA is a developed country.")
obj_ind = India()
obj_usa = USA()
for country in (obj_ind, obj_usa):
    country.capital()
    country.language()
    country.type()
```

17. Polymorphism using overriding

```
from math import pi
class Shape:
    def __init__(self, name):
        self.name = name
    def area(self):
        pass
    def fact(self):
        return "I am a two-dimensional shape."
    def __str__(self):
        return self.name
class Square(Shape):
    def __init__(self, length):
        super().__init__("Square")
        self.length = length
    def area(self):
        return self.length**2
    def fact(self):
        return "Squares have each angle equal to 90 degrees."
class Circle(Shape):
    def __init__(self, radius):
        super().__init__("Circle")
        self.radius = radius
    def area(self):
        return pi*self.radius**2
a = Square(4)
b = Circle(7)
print(b)
print(b.fact())
print(a.fact())
print(b.area())
```

18. Pyqt5

```
from tkinter import *
root = Tk()
myLabel = Label(root, text="hello sir..!")
myLabel.pack()
root.mainloop()
```

19. Single inheritance

```
# Base class
class Parent:
    def func1(self):
        print("This function is in parent class.")
# Derived class
class Child(Parent):
    def func2(self):
        print("This function is in child class.")
# Driver's code
object = Child()
object.func1()
object.func2()
```

20. Sum of even elements

```
sum=0
for i in range(30):
    if i % 2 == 0:
        sum = sum + i
print("Sum of all even elements is : ", sum)
```

21. Thread synchronization

```
import threading
x = 0
def increment():
    global x
    x += 1
def thread_task():
    for _ in range(100000):
        increment()
def main_task():
    global x
    x = 0
    t1 = threading.Thread(target=thread_task)
    t2 = threading.Thread(target=thread_task)
    t1.start()
    t2.start()
    t1.join()
    t2.join()
if __name__ == "__main__":
    for i in range(10):
        main_task()
        print("Iteration {0}: x = {1}".format(i,x))
```

22. Sum of elements using recursion

```
num = 576

def Sum(num):
    if num == 0:
        return 0
    return int(num % 10) + Sum(num / 10)


print(Sum(num))
```