```
#  1 first program
print('Hello, world')
```

```
Hello, world
```

```
#  2 print statements
print("I","love",'python')
print('python','is','fun',sep="-")
print("Hello",end=" ")
print("welcome")
```

```
I love python
python-is-fun
Hello welcome
```

```
# 3 print statements
print("hello\nworld")
print('column 1\tcolumn 2')
```

```
hello
world
column 1        column 2
```

```
# 4 variable and data type
x=10
x="sunki"
print(x)
print(type(x))
```

```
sunki
<class 'str'>
```

```
# 5 printing of personal details
age=21 #int
height=5.7 #float
name="bunny" #str
is_student=True #bool
a=int(10)
b=float(2.7)
print(age)
```

```
21
```

```
# 6 print personal details
name="Bunny"
age=21
course="EEE"
cgpa=9.68
print("NAME:   ",name)
print("AGE:    ",age)
print("COURSE: ",course)
print("CGPA:   ",cgpa)
```

```
NAME:    Bunny
AGE:     21
COURSE:  EEE
CGPA:    9.68
```

```
# 7 print personal details with only one print statement
name=input('enter your name: ')
age=input("enter your age: ")
course=input('enter your course: ')
cgpa=input("enter your CGPA: ")
print(f"\nstudent info: \nNAME: {name}\nAGE: {age}\nCOURSE: {course}\nCGPA: {cgpa}")
```

```
enter your name: sabeera
enter your age: 21
enter your course: EEE
enter your CGPA: 10.

student info:
NAME: sabeera
```

```
AGE: 21
COURSE: EEE
CGPA: 10.
```

```
# 8 print multiplication table

for i in range(1,11):
  for j in range(1,11):
    print(j,"X",i,"=",i*j,end="\t")
  print()
print("\n")
for l in range(1,11):
  for k in range(11,21):
    print(k,"X",l,"=",l*k,end="\t")
  print()
```

```
1 X 1 = 1	2 X 1 = 2	3 X 1 = 3	4 X 1 = 4	5 X 1 = 5	6 X 1 = 6	7 X 1 = 7	8 X 1 = 8	9 X 
1 X 2 = 2	2 X 2 = 4	3 X 2 = 6	4 X 2 = 8	5 X 2 = 10	6 X 2 = 12	7 X 2 = 14	8 X 2 = 16	9 X 
1 X 3 = 3	2 X 3 = 6	3 X 3 = 9	4 X 3 = 12	5 X 3 = 15	6 X 3 = 18	7 X 3 = 21	8 X 3 = 24	9 X 
1 X 4 = 4	2 X 4 = 8	3 X 4 = 12	4 X 4 = 16	5 X 4 = 20	6 X 4 = 24	7 X 4 = 28	8 X 4 = 32	9 X 
1 X 5 = 5	2 X 5 = 10	3 X 5 = 15	4 X 5 = 20	5 X 5 = 25	6 X 5 = 30	7 X 5 = 35	8 X 5 = 40	9 X 
1 X 6 = 6	2 X 6 = 12	3 X 6 = 18	4 X 6 = 24	5 X 6 = 30	6 X 6 = 36	7 X 6 = 42	8 X 6 = 48	9 X 
1 X 7 = 7	2 X 7 = 14	3 X 7 = 21	4 X 7 = 28	5 X 7 = 35	6 X 7 = 42	7 X 7 = 49	8 X 7 = 56	9 X 
1 X 8 = 8	2 X 8 = 16	3 X 8 = 24	4 X 8 = 32	5 X 8 = 40	6 X 8 = 48	7 X 8 = 56	8 X 8 = 64	9 X 
1 X 9 = 9	2 X 9 = 18	3 X 9 = 27	4 X 9 = 36	5 X 9 = 45	6 X 9 = 54	7 X 9 = 63	8 X 9 = 72	9 X 
1 X 10 = 10	2 X 10 = 20	3 X 10 = 30	4 X 10 = 40	5 X 10 = 50	6 X 10 = 60	7 X 10 = 70	8 X 10 = 80	9 X 


11 X 1 = 11	12 X 1 = 12	13 X 1 = 13	14 X 1 = 14	15 X 1 = 15	16 X 1 = 16	17 X 1 = 17	18 X 1 = 18	19 X 
11 X 2 = 22	12 X 2 = 24	13 X 2 = 26	14 X 2 = 28	15 X 2 = 30	16 X 2 = 32	17 X 2 = 34	18 X 2 = 36	19 X 
11 X 3 = 33	12 X 3 = 36	13 X 3 = 39	14 X 3 = 42	15 X 3 = 45	16 X 3 = 48	17 X 3 = 51	18 X 3 = 54	19 X 
11 X 4 = 44	12 X 4 = 48	13 X 4 = 52	14 X 4 = 56	15 X 4 = 60	16 X 4 = 64	17 X 4 = 68	18 X 4 = 72	19 X 
11 X 5 = 55	12 X 5 = 60	13 X 5 = 65	14 X 5 = 70	15 X 5 = 75	16 X 5 = 80	17 X 5 = 85	18 X 5 = 90	19 X 
11 X 6 = 66	12 X 6 = 72	13 X 6 = 78	14 X 6 = 84	15 X 6 = 90	16 X 6 = 96	17 X 6 = 102	18 X 6 = 108	19 X 
11 X 7 = 77	12 X 7 = 84	13 X 7 = 91	14 X 7 = 98	15 X 7 = 105	16 X 7 = 112	17 X 7 = 119	18 X 7 = 126	19 X 
11 X 8 = 88	12 X 8 = 96	13 X 8 = 104	14 X 8 = 112	15 X 8 = 120	16 X 8 = 128	17 X 8 = 136	18 X 8 = 144	19 X 
11 X 9 = 99	12 X 9 = 108	13 X 9 = 117	14 X 9 = 126	15 X 9 = 135	16 X 9 = 144	17 X 9 = 153	18 X 9 = 162	19 X 
11 X 10 = 110	12 X 10 = 120	13 X 10 = 130	14 X 10 = 140	15 X 10 = 150	16 X 10 = 160	17 X 10 = 170	18 X 10 = 180	19 X 
```

```
# 9 print a square number series upto given number
a=int(input("enter a num "))
for i in range(0,a+1):
  print(f"the square of {i} is {i*i}")
```

```
enter a num 8
the square of 0 is 0
the square of 1 is 1
the square of 2 is 4
the square of 3 is 9
the square of 4 is 16
the square of 5 is 25
the square of 6 is 36
the square of 7 is 49
the square of 8 is 64
```

```
# 10 printing of given series 1,3,5,7,9,11
for i in range(1,7):
  print(2*i-1)
```

```
1
3
5
7
9
11
```

```
# 11 arithematic operator
a=10
b=3
print("Addition:",a+b)
print("Subtraction:",a-b)
print("Multiplication:",a*b)
print("Division:",a/b)
print("Floor Division:",a//b)
```

```
print("modulus:",a%b)
print("Exponent:",a**b)
```

```
Addition: 13
Subtraction: 7
Multiplication: 30
Division: 3.3333333333333335
Floor Division: 3
modulus: 1
Exponent: 1000
```

```
# 12 comparision operators
x=5
y=8
print("x==y:",x==y)
print("x!=y;",x!=y)
print("x>y:",x>y)
print("x<y:",x<y)
print("x>=y:",x>=y)
print("x<=y:",x<=y)
```

```
x==y: False
x!=y; True
x>y: False
x<y: True
x>=y: False
x<=y: True
```

```
# 13 logical operators
p=True
q=False
print("p and q:",p and q)
print("p or q",p or q)
print("not p:",not p)
```

```
p and q: False
p or q True
not p: False
```

```
# 14 bitwise operators
m=5
n=3
print("m & n:",m & n)
print("m | n:",m | n)
print("m ^ n:",m ^ n)
print("~m:",~m)
print("m<<1:",m<<1)
print("m>>1:",m>>1)
```

```
m & n: 1
m | n: 7
m ^ n: 6
~m: -6
m<<1: 10
m>>1: 2
```

```
# 15 assignment operators
num=10
num+=5
print("after+=:",num)
num-=3
print("after-=:",num)
num*=2
print("after*=:",num)
num/=4
print("after/=:",num)
```

```
after+=: 15
after-=: 12
after*=: 24
after/=: 6.0
```

```
# 16 membership operators
fruits=("apple","banana","cherry")
```

```
print("banana in fruits:","banana"in fruits)
print("mango not in fruits:","mango"not in fruits)
```

```
banana in fruits: True
mango not in fruits: True
```

```
# 17 identity operator
a=[1,2]
b=[1,2]
c=a
print("a==b",a==b)
print("a is b",a is b)
print("a is c",a is c)
print("a is not b",a is not b)
```

```
a==b True
a is b False
a is c True
a is not b True
```

```
  # 18 greet using function
  def greet(name):
    print("Hello,",name)
  greet("bunny")
  greet("tagore")
```

```
Hello, bunny
Hello, tagore
```

```
# 19 addition using function
def add_numbers(a,b):
  return(a+b)
result=add_numbers(5,7)
print("sum is",result)
```

```
sum is 12
```

```
# 20 list
fruits=["apple","banana","cherry"]
print("fruits:",fruits)
print('first fruit:',fruits[0])
fruits.append("mango")
fruits.remove("banana")
print("updated fruits:",fruits)
```

```
fruits: ['apple', 'banana', 'cherry']
first fruit: apple
updated fruits: ['apple', 'cherry', 'mango']
```

```
# 21 function + list together
def print_fruits(fruits_list):
  for fruit in fruits_list:
    print("fruit:",fruit)
fruits=("apple","banana","cherry")
print_fruits(fruits)
```

```
fruit: apple
fruit: banana
fruit: cherry
```

```
# 22 double the elements of list
def double_numbers(numbers):
  double=[]
  for n in numbers:
    double.append(n*2)
  return double
nums=[1,2,3,4,5]
result=double_numbers(nums)
print("original:",nums)
print("doubled:",result)
```

```
original: [1, 2, 3, 4, 5]
doubled: [2, 4, 6, 8, 10]
```

```
# 23 integrated program
student=["bunny","tagore","muzzameel"]
marks=[(85,90,88),(78,81,86),(92,95,91)]
def total_and_avg(scores):
  total=sum(scores)
  avg=total/len(scores)
  return total,avg
for i in range(len(student)):
  total,avg=total_and_avg(marks[i])
  print(f"{student[i]}:total={total},average={avg:.2f}")
```

```
bunny:total=263,average=87.67
tagore:total=245,average=81.67
muzzameel:total=278,average=92.67
```

```
# 24 printing coordinates using tuple
coordinates=(10,20)
x,y=coordinates
print(x,y)
```

```
10 20
```

```
# 25 square of first 10 numbers
squares=[x**2 for x in range(10)]
print(squares)
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
```

```
# 26 list functions
fruits=["apple","banana","cherry"]
fruits.append("orange")
fruits.insert(1,"grape")
fruits[2]="blue berry"
print(fruits)
```

```
['apple', 'grape', 'blue berry', 'cherry', 'orange']
```

```
# 27 addition using function
def add(a,b=0):
  return(a+b)
print(add(5))
print(add(3,7))
print(add(b=2,a=8))
```

```
5
10
10
```

```
# 28 greet using function
def greet(name):
 print(f"hello,{name}!")
greet("bunny")
```

```
hello,bunny!
```

```
# 29 diploma percentage
def diploma_percentage(a,b,c,d,e):
  return((((a*.25)+b+c+d+e)/3650)*100)
a=int(input("enter your first sem marks: "))
b=int(input("enter your third sem marks: "))
c=int(input("enter your fourth sem marks: "))
d=int(input("enter your fifth sem marks: "))
e=int(input("enter your sixth sem marks: "))
t=(a*.25)+b+c+d+e
print(f"your total marks is:{t}/3650")
y=diploma_percentage(a,b,c,d,e)
print("your diploma percentage is",y)
if y>=90:
  print("Your grade is-A")
elif y>=80:
  print("your grade is-B")
elif y>=70:
  print("your grade is-C")
elif y>=60:
```

```
  print("your grade is-D")
elif y>=50:
  print("your grade is-E")
else:
  print("your grade is-FAIL")
```

```
enter your first sem marks: 709
enter your third sem marks: 728
enter your fourth sem marks: 730
enter your fifth sem marks: 881
enter your sixth sem marks: 286
your total marks is:2802.25/3650
your diploma percentage is 76.77397260273973
your grade is-C
```

```
# 30 simple addition
def add_numbers(a,b):
  return(a+b)
result=add_numbers(3,5)
print("the sum is:",result)
```

```
the sum is: 8
```

```
# 31 check even or odd
def check_even_odd(num):
  if num%2==0:
    print(f"{num} is even")
  else:
    print(f"{num} is odd")
num=int(input("enter a number: "))
check_even_odd(num)
```

```
enter a number: 8
8 is even
```

```
# 32 append to a list
num=[1,2,3,4,5]
num.append(6)
print("updated list is:",num)
```

```
updated list is: [1, 2, 3, 4, 5, 6]
```

```
# 33 iterative over a list
colours=["red","green","blue"]
for colour in colours:
  print("colour:",colour)
```

```
colour: red
colour: green
colour: blue
```

```
# 34 creat and print a tuple
person=("bunny",21,"gooty")
print(f"person tuple:{person}")
```

```
person tuple:('bunny', 21, 'gooty')
```

```
# 35 access elements in tuple
num=(10,20,30)
print("first element:",num[0])
print(f"last element: {num[-1]}")
```

```
first element: 10
last element: 30
```

```
# 36 ohm's law
def ohms_law(voltage=None,current=None,resistance=None):
  if voltage is None:
    return current*resistance
  elif current is None:
    return voltage/resistance
  elif resistance is None:
    return voltage/current
v=ohms_law(current=2,resistance=5)
```

```
print("voltage:",v,"v")
i=ohms_law(voltage=10,resistance=5)
print("current:",i,"A")
r=ohms_law(voltage=10,current=2)
print("resistance:",r,"ohm")
```

```
voltage: 10 v
current: 2.0 A
resistance: 5.0 ohm
```

```
# 37 power claculation
v=250
i=10
p=v*i
print(f"the power is {p}watts")
```

```
the power is 2500watts
```

```
# 38 stress
force=500
area=0.005
stress=force/area
print(f"the stress is {stress}pa")
```

```
the stress is 100000.0pa
```

```
# 39 series resistance
resistors=[4,6,10]
total_resistance=sum(resistors)
print(f"the total resistance is {total_resistance}ohm")
```

```
the total resistance is 20ohm
```

```
# 40 area of a rectangle
def area_rectangle(length,width):
  return length*width
l=float(input("enter the length: "))
w=float(input("enter the width: "))
a=area_rectangle(l,w)
print(f"the area of the rectangle is {a}m^2")
```

```
enter the length: 10
enter the width: 5
the area of the rectangle is 50.0m^2
```

```
# 41 volume of concrte
def volume_cubiod(length,width,height):
  return length*width*height
length=0.5
width=0.5
height=3
v=volume_cubiod(length,width,height)
print("the volume of the cubiod is ",v,"m^3")
```

```
the volume of the cubiod is  0.75 m^3
```

```
# 42 stress calculation usin function
def stress(force,area):
  return force/area
s=stress(1000,0.02)
print(f"the stress is {s}pa")
```

```
the stress is 50000.0pa
```

```
# 43 Area of a circle
import math
def area_circle(radius):
  return math.pi*radius**2
r=float(input("enter the radius: "))
a=area_circle(r)
print(f"the area of the circle is {a}m^2")
```

```
enter the radius: 5
the area of the circle is 78.53981633974483m^2
```

```
# 44 torque calculation
force=50
distance=.2
torque=force*distance
print(f"the torque is {torque}N*m")
```

```
the torque is 10.0N*m
```

```
# 45 kinetic energy
m=1000
v=20
ke=0.5*m*v**2
print(f"kinetic energy: {ke}joules")
```

```
kinetic energy: 200000.0joules
```

```
# 46 celsius to fahrenheit
def c_to_f(c):
  return (c*9/5)+32
c=float(input("enter the temperature in celsius: "))
f=c_to_f(c)
print(f"the temperature in fahrenheit is {f}f")
```

```
enter the temperature in celsius: 36
the temperature in fahrenheit is 96.8f
```

```
# 47 efficiency calculation
output_power=800
input_power=1000
efficiency=(output_power/input_power)*100
print(f"the efficiency is {efficiency}%")
```

```
the efficiency is 80.0%
```

```
# 48 displacement calculation
import math
a=5
omega=2*math.pi
t=0.5
x=a*math.cos(omega*t)
print("displacement:",x,"m")
```

```
displacement: -5.0 m
```

```
# 49 elgibility check for voting
age=int(input('ener your age:'))
if age>18:
  print('you are eligible to vote')
else:
  print('you are not eligible to vote')
```

```
ener your age:21
you are eligible to vote
```

```
# 50 print given series 4,8,16,32
for i in range(2,6):
  print(2**i)
```

```
4
8
16
32
```

```
# 51 printing personal details
name=input("enter your name: ")
age=int(input("enter your age: "))
print("name:",name,"you are",age)
```

```
enter your name: sabeera
enter your age: 21
```

```
name: sabeera you are 21
```

```
# 52 grade of a student
marks=int(input("enter your marks"))
if marks>=90:
  print(" Grade A")
elif marks>=60:
  print("Grade B")
else:
  print("Grade C")
```

```
enter your marks96
 Grade A
```

```
# 53 python program using class
class student:
  def __init__(self,name,age,marks):
    self.name=name
    self.age=age
    self.marks=marks
s1=student("bunny",21,96)
print(s1.name,s1.age,s1.marks)
```

```
bunny 21 96
```

```
# 54 dictionary
student={"name":"bunny","age":21,"marks":96}
print(student["name"])
student['marks']=90
print(student)
```

```
bunny
{'name': 'bunny', 'age': 21, 'marks': 90}
```

```
# 55 string operation
text="python is fun"
print(text.upper())
print(text.lower())
print(text.replace("fun","awesome"))
print(text.split())
print(text[0:6])
```

```
PYTHON IS FUN
python is fun
python is awesome
['python', 'is', 'fun']
python
```

```
# 56 for & while loop
for i in range(1,6):
  print(i)
count=1
while count<=5:
  print(count)
  count+=1
```

```
1
2
3
4
5
1
2
3
4
5
```

```
# 57 list
fruits=["apple","banana","cherry"]
fruits.append("orange")
print(fruits)
```

```
['apple', 'banana', 'cherry', 'orange']
```

```
# 58 tuple
dimensions=[10,20,30]
print(dimensions[0])
```

```
10
```

```
# 59 sets
numbers={1,2,3,3}
print(numbers)
```

```
{1, 2, 3}
```

```
# 60 radha krishna
a=input("enter your god name")
for i in range(1,3):
  print(a)
```

```
enter your god nameradha krishna
radha krishna
radha krishna
```

```
# 61 torque calculation
def torque(force,radius):
  return force*radius
print("torque",torque(50,31),"N-M")
```

```
torque 1550 N-M
```

```
# 62 machine info using dictionary
machine={"name":"DC shunt generator ","rating":3,"unit":"KW","speed":1500,"units":"RPM"}
print("machine:",machine["name"])
print("power rating:",machine["rating"],machine["unit"])
print("speed:",machine["speed"],machine["units"])
```

```
machine: DC shunt generator
power rating: 3 KW
speed: 1500 RPM
```

```
# 63 set uniq tools
tools={"wrench","hammer","screwdriver","hammer"}
print("uniq tools:",tools)
```

```
uniq tools: {'wrench', 'screwdriver', 'hammer'}
```

```
# 64 slab volume
def slab_volume(length,width,thickness):
  return length*width*thickness
a=float(input("enter length value: "))
b=float(input("enter width value: "))
c=float(input("enter thickness value : "))
print("the slab volume is: ",slab_volume(a,b,c),"m^3")
```

```
enter length value: 50
enter width value: 45
enter thickness value : 10
the slab volume is:  22500.0 m^3
```

```
# 65 project details using dictionary:
project={"name":"wire less AC power detector","cost":1200,"units":'$',"status":"completed"}
print("The project name is:",project["name"])
print("The project cost is:",project["cost"],project["units"])
print("The project status is:",project["status"])
```

```
The project name is: wire less AC power detector
The project cost is: 1200 $
The project status is: completed
```

```
# 66 area of triangles using loop
bases=[10,15,20]
heights=[5,8,12]
for base,height in zip(bases,heights):
  area=0.5*base*height
  print(f"Area with base {base} and height {height} area: {area}m^2")
```

```
Area with base 10 and height 5 area: 25.0m^2
Area with base 15 and height 8 area: 60.0m^2
Area with base 20 and height 12 area: 120.0m^2
```

```
# 67 ohms law
def voltage(current,resistance):
  return current*resistance
print("voltage:",voltage(2.5,10),"v")
```

```
voltage: 25.0 v
```

```
# 68 set unique components
components={"motor","battery","switch","motor","switch"}
print("unique components:",components)
```

```
unique components: {'battery', 'switch', 'motor'}
```

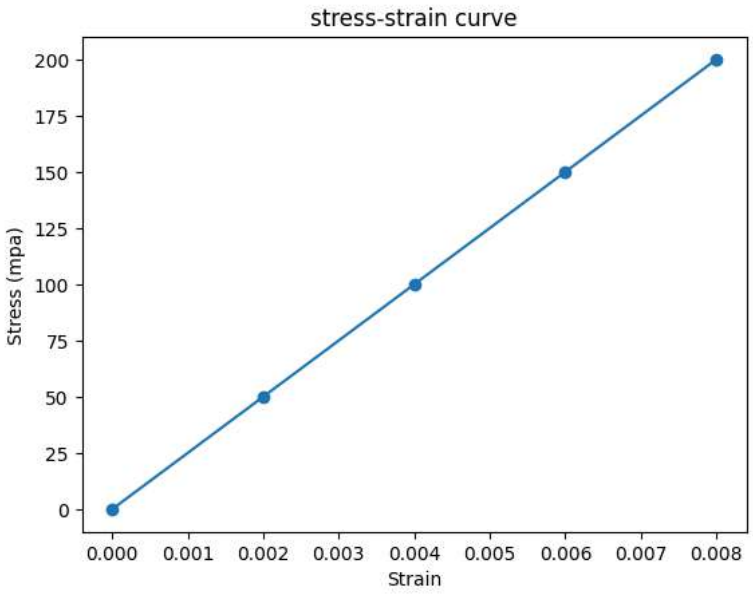
```
# 69 power clculation using loop
voltages=[230,120,12]
current=[5,10,.5]
for v,i in zip(voltages,current):
  power=v*i
  print(f"power for {v}v and {i}A :{power} Watts")
```

```
power for 230v and 5A :1150 Watts
power for 120v and 10A :1200 Watts
power for 12v and 0.5A :6.0 Watts
```

```
# 70 numpy example
import pandas as pd
df=pd.DataFrame({"Beam ID":["B1","B2",],"length (m)":[5,6],"load per m(KN/m)":[10,12]})
df["Total Load"]=df["length (m)"]*df["load per m(KN/m)"]
print(df)
```

```
  Beam ID  length (m)  load per m(KN/m)  Total Load
0      B1           5                10          50
1      B2           6                12          72
```

```
# 71 library matplotlib
import matplotlib.pyplot as plt
Stress=[0,50,100,150,200]
Strain=[0,0.002,0.004,0.006,0.008]
plt.plot(Strain,Stress,marker="o")
plt.xlabel("Strain")
plt.ylabel("Stress (mpa)")
plt.title("stress-strain curve")
plt.show()
```



```
# 72 stress
stress=250
if stress>300:
  print("stress is high")
else:
  print("stress is low")
```

```
stress is low
```

```
# 73 volume of beam
def beam_volume(l,b,h):
  return l*b*h
print("volume:",beam_volume(10,5,2),"m^3")
```

```
volume: 100 m^3
```

```
# 74 loop
for load in[100,200,300]:
  print("load:",load)
count=0
print("\n")
while count<3:
  print("count:",count)
  count+=1
```

```
load: 100
load: 200
load: 300


count: 0
count: 1
count: 2
```

```
# 75 filtering high efficiency
efficiencies=[0.85,0.9,0.92,0.8]
high=list(filter(lambda e:e>=0.88,efficiencies))
print("High efficiencies:",high)
```

```
High efficiencies: [0.9, 0.92]
```

```
# 76 printing using function
def  beam_ids(n):
  for i in range(1,n+1):
    print(f"beam {i}")
beam_ids(5)
```

```
beam 1
beam 2
beam 3
beam 4
beam 5
```

```
# 77 printing hallticket numbers
n=int(input("enter a number: "))
for i in range(1,10):
  print(f"24705A020{i}")
  print()
for j in range(10,n+1):
  print(f'24705A02{j}')
  print()
```

```
enter a number: 80
24705A0201

24705A0202

24705A0203

24705A0204

24705A0205

24705A0206
```

```
24705A0207

24705A0208

24705A0209

24705A0210

24705A0211

24705A0212

24705A0213

24705A0214

24705A0215

24705A0216

24705A0217

24705A0218

24705A0219

24705A0220

24705A0221

24705A0222

24705A0223

24705A0224

24705A0225

24705A0226

24705A0227

24705A0228
```

```
# 78 printing moment
forces=[100,200,300]
distances=[2,4,6]
moments=[f*d for f,d in zip(forces,distances)]
print("moments:",moments)
```

```
moments: [200, 800, 1800]
```

```
#  79 gear rotation
gear_data={"g1":(20,40),"g2":(15,45)}
ratios={k:driven/driving for k,(driven,driving) in gear_data.items()}
print("gear ratios:",ratios)
```

```
gear ratios: {'g1': 0.5, 'g2': 0.3333333333333333}
```

```
# 80 ConcreteSlab
class ConcreteSlab:
    def __init__(self, l, b, h, cost_per_m3):
        self.l = l
        self.b = b
        self.h = h
        self.cost_per_m3 = cost_per_m3
    def volume(self):
        return self.l* self.b* self.h
    def total_cost(self):
        return self.volume()*self.cost_per_m3
slab = ConcreteSlab (10, 5, 0.2, 3500)
print("Volume:", slab.volume(), "m³")
print("Total Cost:", slab.total_cost(), "INR")
```

```
Volume: 10.0 m³
Total Cost: 35000.0 INR
```

```
# 81 density
density = 7850
volume= 0.002
mass=density*volume
print("Mass (kg):", mass)
```

```
Mass (kg): 15.700000000000001
```

```
# 82 3 phase power
import math

class ThreePhasePower:
    def __init__(self, V, I, pf):
        self.V = V
        self.I = I
        self.pf = pf
    def power (self):
        return math.sqrt(3)*self.V * self.I *self.pf
tp=ThreePhasePower (400, 50, 0.9)
print("Three-Phase Power:", round(tp.power(), 2), "W")
```

```
Three-Phase Power: 31176.91 W
```

```
#  83 motor out put power
class Motor:
    def __init__(self,power_kw,efficiency):
      self.power_kw=power_kw
      self.efficiency=efficiency
    def output_power(self):
        return self.power_kw * self.efficiency
Motor1=Motor(10, 0.92)
print(Motor1.output_power())
```

```
9.200000000000001
```

```
# 84 EXCEPTION HANDLING
try:
  data=int(input("enter load value"))
  print('load',data)
except ValueError:
  print("invalid input! Enter a number")
```

```
enter load value8106430181
load 8106430181
```

```
# 85 JSON & APIs
import json
with open('machine_config.json')as f:
  config=json.load(f)
print(config["motor"]["rated_speed"])
```

```
.#  86 SciPy for calculation
from scipy.optimize import fsolve
def eq(x):
  return 3*x**2-x-1
root=fsolve(eq,0)
print("root:",root)
```

```
root: [-0.43425855]
```

```
# 87 monte carlo simulation
import numpy as np
samples = np.random.normal(50,5,10000)
failure_rate=np.mean(samples>60)
print('Failure Rate:',failure_rate*100,"%")
```

```
Failure Rate: 2.15 %
```

```
# 88 Multi - processing
from multiprocessing import Pool
def square(n):
  return n*n
```

```
with Pool() as p:
  print(p.map(square,[1,2,3,4,5]))
```

```
[1, 4, 9, 16, 25]
```

```
# 89 using numpy lib
import numpy as np
from scipy.optimize import curve_fit
def model(x, a, b):
    return a * np.exp(b*x)
x = np.linspace(0, 4, 50)
y = model(x, 2, 0.5) + np.random.normal(0, 0.2, 50)
popt,_=curve_fit(model, x, y)
print(popt)
```

```
[1.9512697  0.50697074]
```

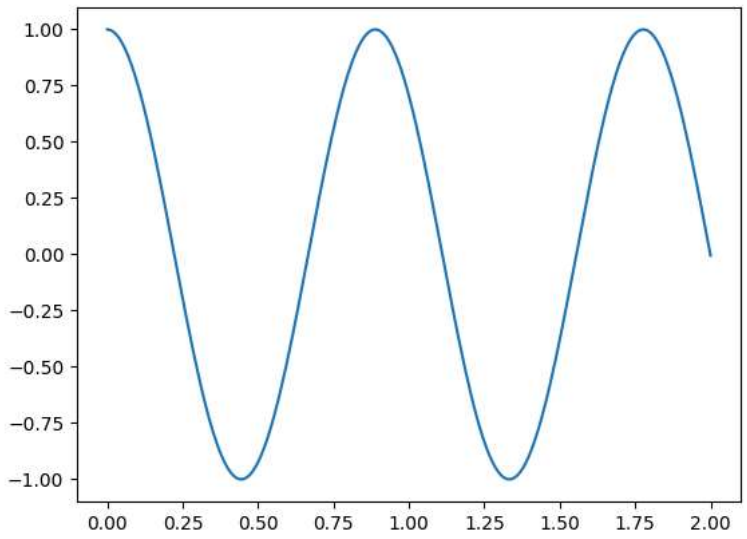
```
# 90
import tkinter as tk
def calc():
    F =float(entry_force.get())
    A =float(entry_area.get())
    result.set(F/A)
root=tk.Tk()
entry_force = tk.Entry(root)
entry_area = tk.Entry(root)
result = tk.StringVar()
tk.Button(root,text='Calc',command=calc).pack()
tk.Label(root,textvariable=result).pack()
root.mainloop()
print(loop)
```

```
# 91 using pandas lib
import pandas as pd
df = pd.read_excel('data.xlsx')
df['Stress'] = df['Force'] / df['Area']
df.to_excel('results.xlsx', index=False)
```

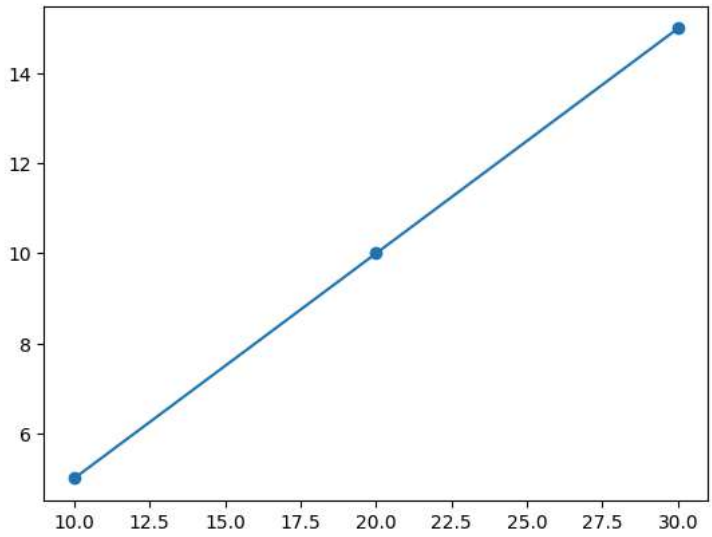
```
#  92 heat trasfer example
k=205; A = 0.01; T1, T2 = 373, 293; L = 0.05
Q = (k*A*(T1-T2))/L
print(Q)
```

```
3280.0
```

```
#  93 vibration example
import matplotlib.pyplot as plt
m,k = 1.0, 50.0
omega = np.sqrt(k/m)
t = np.linspace(0, 2, 200)
x = np.cos(omega *t)
plt.plot(t,x);plt.show()
```



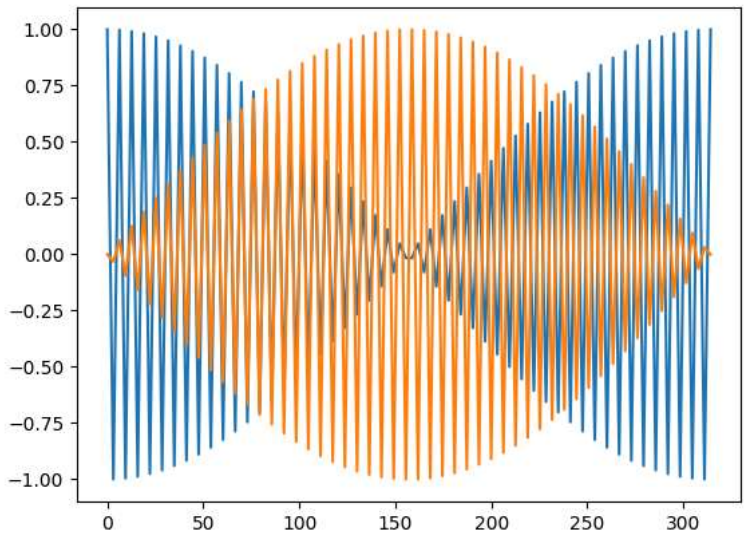

```
# 94 v-i relation
import matplotlib.pyplot as plt
import numpy as np
r=(2,2,2)
y=(10,20,30)
x=np.divide(y,r)
plt.plot(y,x,marker="o");
plt.show()
```




```
# 95 bending moment
w,l=5,6
M_max=(w*l**2)/8
print(M_max)
```

```
22.5
```

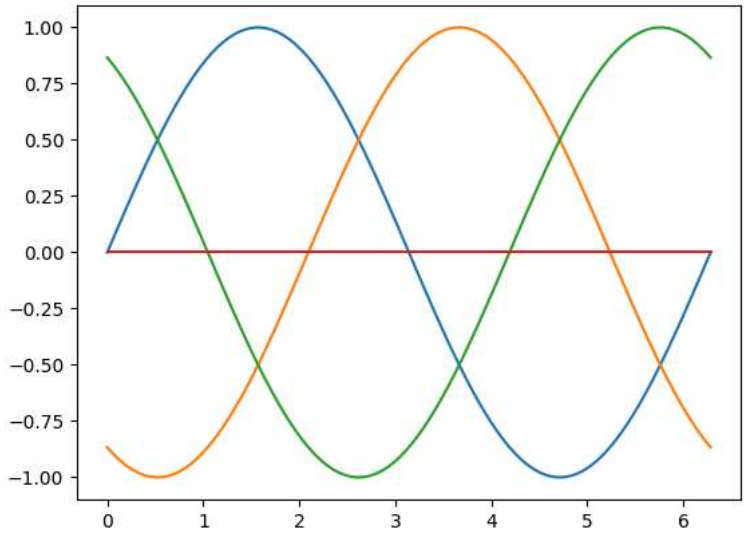
```
# 96  Sine wave program
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 100* np.pi,100)
y = np.cos(x)
x=np.linspace(0,100*np.pi,100)
y1=np.sin(x)
plt.plot(x,y);
plt.plot(x,y1);
plt.show()
```




```
from os.path import commonpath
# 97 ac circuit
import cmath
v=230;z=10+5j
i=v/z
print("the current value is :",i)
```

```
the current value is : (18.4-9.2j)
```

```
# 98 3-phase wavwe forms
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 2* np.pi, 100)
y_ = np.sin(x)
y = np.sin(x-(2*np.pi)/3)
y__=np.sin(x-(4*np.pi)/3)
y1=(x-x)
plt.plot(x, y_)
plt.plot(x, y)
plt.plot(x, y__)
plt.plot(x, y1)
plt.show()
```




```
# 99 array, as set as array
import numpy as np
arr=np.array({1,2,5,4,3,6,4,5})
print(arr)
```

```
{1, 2, 3, 4, 5, 6}
```

```
# 100 array, list as array
import numpy as np
arr=np.array([1,2,5,4,3,6,4,5])
print(arr)
```

```
[1 2 5 4 3 6 4 5]
```

```
# 101 arra, tuple as array
import numpy as np
arr=np.array((1,2,5,4,3,6,4,5))
print(arr)
```

```
[1 2 5 4 3 6 4 5]
```

```
# 102 array, dictionary as array
import numpy as np
arr=np.array({1:1,2:2,5:5,4:4,3:3,6:6,4:4,5:5})
print(arr)
```

```
{1: 1, 2: 2, 5: 5, 4: 4, 3: 3, 6: 6}
```

```
# 103 third year list
thirdyear_list=[10,50,100]
print("original:",thirdyear_list)
thirdyear_list.append(90)
thirdyear_list[1]=80
print("updated:",thirdyear_list)
```

```
original: [10, 50, 100]
updated: [10, 80, 100, 90]
```

```
#  104 diploma marks
dip=[855,901,941,1049,299]
to=[1000,1000,1000,1100,300]
print(f"First sem marks is {dip[0]}")
print(f"third sem marks is {dip[1]}")
print(f"fourth sem marks is {dip[2]}")
print(f"fifth sem marks is {dip[3]}")
print(f"sixth sem marks is {dip[4]}")
print(f"total marks is {sum(dip)}")
print(f"persentage marks is {(sum(dip)/sum(to))*100}")
```

```
First sem marks is 855
third sem marks is 901
fourth sem marks is 941
fifth sem marks is 1049
sixth sem marks is 299
total marks is 4045
persentage marks is 91.93181818181819
```

```
#  105 adding two tuples
student_tuple=(1,2,3,4,5,6,7,8,9,10)
print("original",student_tuple)
new_tuple=student_tuple+(11,12,13)
print("new tuple",new_tuple)
```

```
original (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
new tuple (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13)
```

```
# 106 tuple slicing
student_tuple=(1,2,3,4,5,6,7,8,9,10)
print("original",student_tuple)
new_tuple=student_tuple[2:9]
print("new tuple",new_tuple)
```

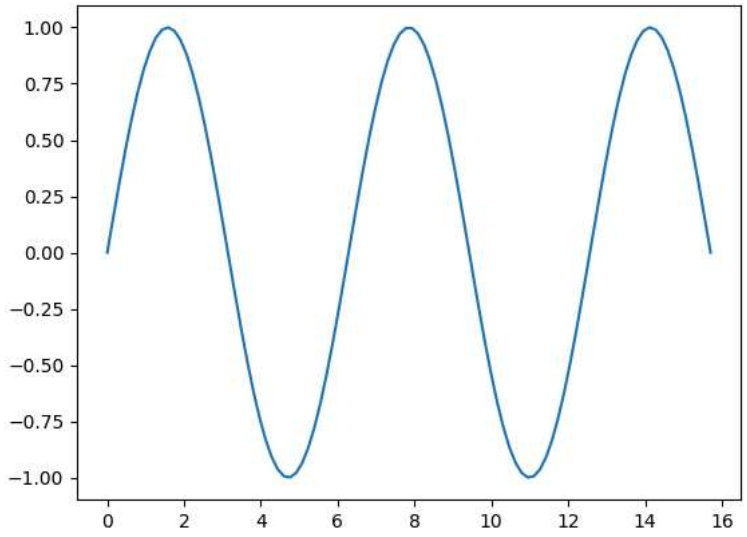
```
original (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
new tuple (3, 4, 5, 6, 7, 8, 9)
```

```
# 107 set operations
student_set={1,2,3,4,5,6,7,8,9,10}
print("original",student_set)
student_set.add(11)
```

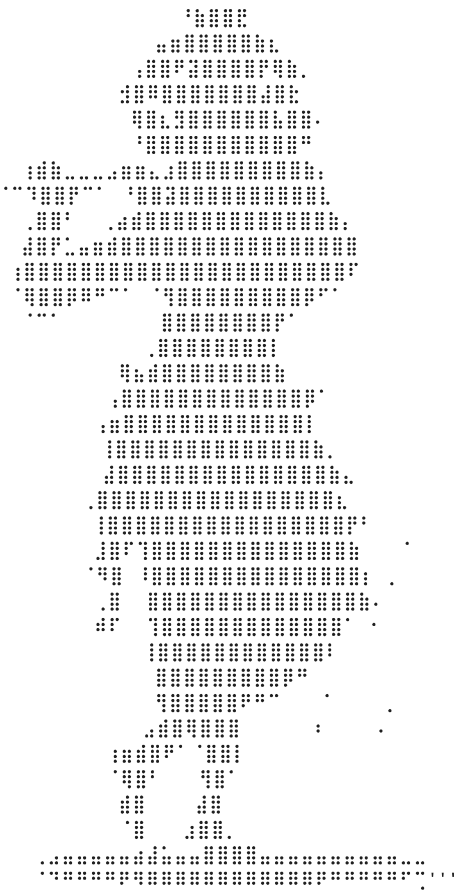
```
student_set.add(12)
student_set.remove(3)
print("updated",student_set)
```

```
original {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
updated {1, 2, 4, 5, 6, 7, 8, 9, 10, 11, 12}
```

```
# 108 sigle phase wave form
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,5*np.pi,100)
y=np.sin(x)
plt.plot(x,y)
plt.show()
```




```
# 109 printing krishna image
a='''
```

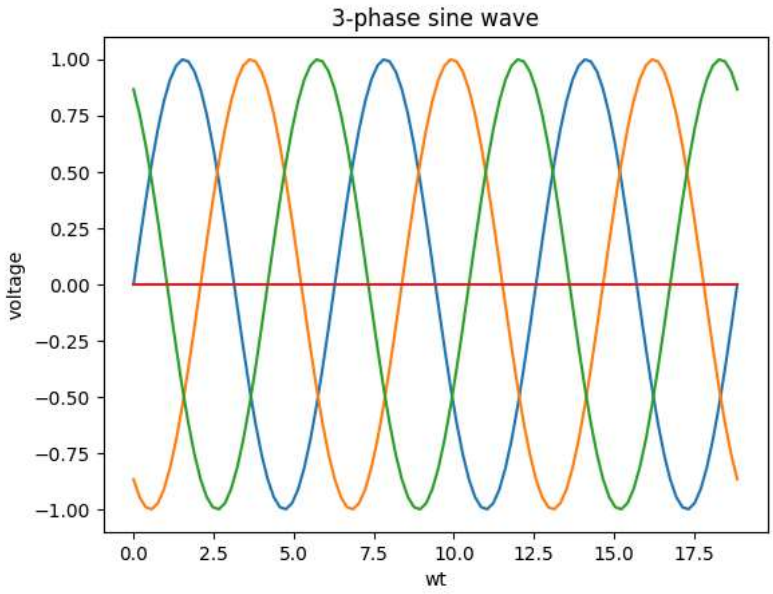


```
print(a)
```

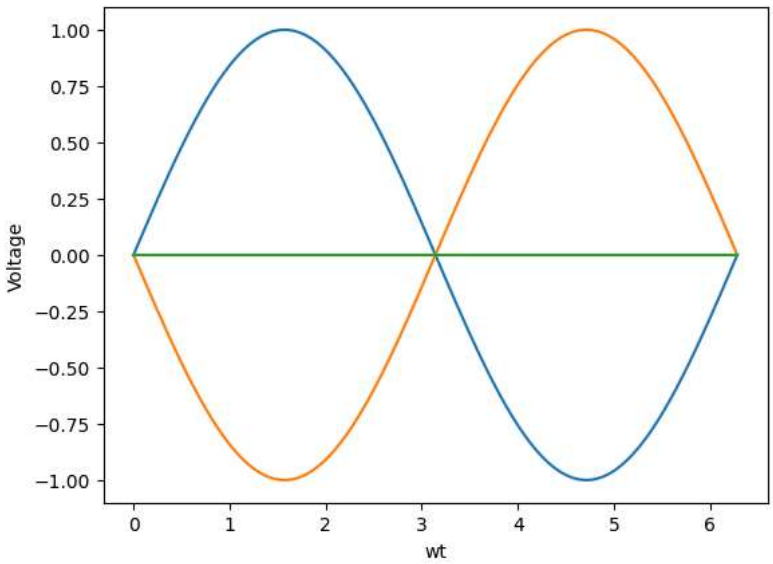
```
# 110 printing address using class
class address:
  def __init__(self,street,city,state):
    self.street=street
    self.city=city
    self.state=state
a=input("Enter your street:")
b=input("Entyer your city:")
c=input("Enter your state:")
s1=address(a,b,c)
print(s1.street)
print(s1.city)
print(s1.state)
```

```
Enter your street:oc colony
Entyer your city:lachanapali
Enter your state:ap
oc colony
lachanapali
ap
```

```
# 111 3-phase wave forms
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,6*np.pi,100)
y=np.sin(x)
y1=np.sin(x-(2*np.pi)/3)
y2=np.sin(x-(4*np.pi)/3)
y3=np.sin(x-x)
plt.plot(x,y)
plt.plot(x,y1)
plt.plot(x,y2)
plt.plot(x,y3)
plt.xlabel("wt")
plt.ylabel("voltage")
plt.title("3-phase sine wave")
plt.show()
```




```
# 114 print given wave form
import matplotlib.pyplot as plt
import numpy as np
x=np.linspace(0,2*np.pi,100)
y=np.sin(x)
y1=np.sin(-x)
y2=np.sin(x-x)
plt.plot(x,y)
plt.plot(x,y1)
plt.plot(x,y2)
plt.xlabel("wt")
plt.ylabel("Voltage")
plt.show()
```




```
matrix_a=np.array([[1,2],[7,8]])
matrix_b=np.array([[3,4],[5,6]])
add=np.multiply(matrix_a,matrix_b)
print(add)
```

```
[[ 3  8]
 [35 48]]
```

```
# final project
import numpy as np
```

```
import matplotlib.pyplot as plt
from scipy.signal import firwin, lfilter, freqz, butter, filtfilt

def design_fir_filter(order, cutoff, fs, filter_type='low'):
    nyq = 0.5 * fs
    normalized_cutoff = np.array(cutoff) / nyq

    if filter_type == 'low':
        pass_zero = True
    elif filter_type == 'high':
        pass_zero = False
    elif filter_type in ['bandpass', 'bandstop']:
        pass_zero = filter_type
    else:
        raise ValueError("Invalid filter_type. Choose from 'low', 'high', 'bandpass', 'bandstop'.")

    fir_coeff = firwin(order + 1, normalized_cutoff, pass_zero=pass_zero)
    return fir_coeff

def design_iir_filter(order, cutoff, fs, filter_type='low'):
    nyq = 0.5 * fs
    normalized_cutoff = np.array(cutoff) / nyq
    b, a = butter(order, normalized_cutoff, btype=filter_type, analog=False)
    return b, a

def apply_filter(b, a, signal, use_filtfilt=False):
    if use_filtfilt:
        filtered_signal = filtfilt(b, a, signal)  # zero-phase filtering
    else:
        filtered_signal = lfilter(b, a, signal)
    return filtered_signal

def plot_frequency_response(b, a=1, fs=1.0, title='Frequency Response'):
    w, h = freqz(b, a, worN=8000)
    plt.plot((fs * 0.5 / np.pi) * w, 20 * np.log10(abs(h)), 'b')
    plt.title(title)
    plt.xlabel('Frequency (Hz)')
    plt.ylabel('Gain (dB)')
    plt.grid(True)
    plt.show()

if __name__ == "__main__":
    fs = 500.0       # Sampling frequency (Hz)
    t = np.arange(0, 2.0, 1/fs)  # Time vector for 2 seconds

    # Create a noisy signal: 5 Hz sine + white noise + 50 Hz interference
    freq_signal = 5.0
    noisy_signal = (
        np.sin(2 * np.pi * freq_signal * t) +
        0.5 * np.random.randn(len(t)) +
        0.3 * np.sin(2 * np.pi * 50 * t)
    )

    # Filter parameters
    fir_order = 50
    iir_order = 4
    cutoff = 10  # cutoff frequency (Hz)
    filter_type = 'low'  # lowpass filter

    # Design FIR filter
    fir_b = design_fir_filter(fir_order, cutoff, fs, filter_type)

    # Design IIR filter
    iir_b, iir_a = design_iir_filter(iir_order, cutoff, fs, filter_type)

    # Apply filters
    fir_filtered = lfilter(fir_b, [1.0], noisy_signal)
    iir_filtered = apply_filter(iir_b, iir_a, noisy_signal, use_filtfilt=True)

    # Plot time domain signals
    plt.figure(figsize=(12, 8))

    plt.subplot(3, 1, 1)
    plt.plot(t, noisy_signal, label='Noisy Signal')
    plt.legend()
    plt.xlabel('Time [s]')
    plt.ylabel('Amplitude')
```

```
plt.subplot(3, 1, 2)
plt.plot(t, fir_filtered, label='FIR Filtered Signal', color='orange')
plt.legend()
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')

plt.subplot(3, 1, 3)
plt.plot(t, iir_filtered, label='IIR Filtered Signal', color='green')
plt.legend()
plt.xlabel('Time [s]')
plt.ylabel('Amplitude')

plt.tight_layout()
plt.show()

# Plot frequency responses
plot_frequency_response(fir_b, fs=fs, title='FIR Filter Frequency Response')
plot_frequency_response(iir_b, iir_a, fs=fs, title='IIR Filter Frequency Response')
```