1. **Event Reminder Application**
   - Your task is to create an event reminder application. The application should accept the event name and event date (in `YYYY-MM-DD` format) from the user and calculate how many days are left until the event.

```
import datetime

def days_until_event(event_name, event_date):
    # Parse the event date
    event_date = datetime.datetime.strptime(event_date, '%Y-%m-
%d').date()

    # Get the current date
    current_date = datetime.date.today()
    print(f"Current time is {current_date}")

    # Calculate the difference between the event date and the current
date
    delta = event_date - current_date

    # Return the number of days until the event
    return delta.days

# Get the event name and date from the user
event_name = input("Enter the event name: ")
event_date = input("Enter the event date (YYYY-MM-DD): ")

# Calculate the number of days until the event
days_until = days_until_event(event_name, event_date)

# Print the number of days until the event
if days_until > 0:
    print(f"There are {days_until} days until {event_name}.")
elif days_until == 0:
    print(f"Today is the day of {event_name}!")
else:
    print(f"{event_name} has already passed.")
```

```
Enter the event name: golf
Enter the event date (YYYY-MM-DD): 2024-11-12
Current time is 2024-06-18
There are 147 days until golf.
```

1. **Time Duration Calculation**
   - Write a program that asks the user to input two times (start time and end time) in `HH:MM:SS` format. Calculate and print the duration

```python
# prompt: 2. **Time Duration Calculation**
#    - Write a program that asks the user to input two times (start time and end time) in `HH:MM:SS` format. Calculate and print the duration b

def calculate_duration(start_time, end_time):
  # Convert the start and end times to datetime objects
  start_time = datetime.datetime.strptime(start_time, '%H:%M:%S')
  end_time = datetime.datetime.strptime(end_time, '%H:%M:%S')

  # Calculate the difference between the end time and the start time
  delta = end_time - start_time

  # Extract the hours, minutes, and seconds from the timedelta object
  hours = delta.seconds // 3600
  minutes = (delta.seconds // 60) % 60
  seconds = delta.seconds % 60

  # Print the duration
  print(f"Duration: {hours:02d}:{minutes:02d}:{seconds:02d}")

# Get the start and end times from the user
start_time = input("Enter the start time (HH:MM:SS): ")
end_time = input("Enter the end time (HH:MM:SS): ")

# Calculate and print the duration
calculate_duration(start_time, end_time)
```

```
Enter the start time (HH:MM:SS): 2:30:00
Enter the end time (HH:MM:SS): 4:30:00
Duration: 02:00:00
```

1. **Date of Birth and Age Calculation**
   - Create a program that asks the user for their date of birth in `YYYY-MM-DD` format and calculates their current age in years, months, and days.

```python
# prompt: 3. **Date of Birth and Age Calculation**
#    - Create a program that asks the user for their date of birth in `YYYY-MM-DD` format and calculates their current age in years, months, and days.

from datetime import datetime

def calculate_age(birth_date):
  # Get the current date
  current_date = datetime.today()

  # Parse the birth date
  birth_date = datetime.strptime(birth_date, '%Y-%m-%d')
```

```
  # Calculate the difference between the current date and the birth
date
  delta = current_date - birth_date

  # Calculate the years, months, and days
  years = delta.days // 365
  months = (delta.days % 365) // 30
  days = (delta.days % 365) % 30

  # Print the age
  print(f"You are {years} years, {months} months, and {days} days
old.")

# Get the date of birth from the user
birth_date = input("Enter your date of birth (YYYY-MM-DD): ")

# Calculate and print the age
calculate_age(birth_date)
```

```
Enter your date of birth (YYYY-MM-DD): 2004-12-4
You are 19 years, 6 months, and 21 days old.
```

1. **Weekly Report Generator**
   - Write a script that generates a weekly report file named `report_<date>.txt`, where `<date>` is the current date. The file should contain a timestamp of when the report was generated.

```
# prompt: 4. **Weekly Report Generator**
#    - Write a script that generates a weekly report file named
`report_<date>.txt`, where `<date>` is the current date. The file
should contain a timestamp of when the report was generated.

import datetime

# Get the current date
today = datetime.date.today()

# Format the date for the file name
date_str = today.strftime("%Y-%m-%d")

# Create the file name
file_name = f"report_{date_str}.txt"

# Open the file for writing
with open(file_name, "w") as f:
  # Get the current time
  now = datetime.datetime.now()

  # Write the timestamp to the file
  f.write(f"Report generated on: {now}\n")
```

1. **Project Deadline Tracker**
   - Develop a program to track project deadlines. The user should be able to enter the project name and the deadline date. The program should display a message if the deadline is within 7 days from the current date.

```
# prompt: 5. **Project Deadline Tracker**
#    - Develop a program to track project deadlines. The user should be able to enter the project name and the deadline date. The program should display a message if the deadline is within 7 days from the current date.

from datetime import datetime, timedelta

def track_deadline(project_name, deadline_date):
  # Parse the deadline date
  deadline_date = datetime.strptime(deadline_date, '%Y-%m-%d')

  # Get the current date
  current_date = datetime.today()

  # Calculate the difference between the deadline date and the current date
  delta = deadline_date - current_date

  # Check if the deadline is within 7 days
  if delta <= timedelta(days=7):
    print(f"WARNING: The deadline for {project_name} is within 7 days!")
  else:
    print(f"The deadline for {project_name} is {delta.days} days away.")

# Get the project name and deadline date from the user
project_name = input("Enter the project name: ")
deadline_date = input("Enter the deadline date (YYYY-MM-DD): ")

# Track the deadline
track_deadline(project_name, deadline_date)
```

1. **Meeting Scheduler**
   - Create a meeting scheduler program that accepts the meeting date and time (in `YYYY-MM-DD HH:MM` format) from the user and calculates the number of days, hours, and minutes left until the meeting.

```
# prompt: 6. **Meeting Scheduler**
#    - Create a meeting scheduler program that accepts the meeting date and time (in `YYYY-MM-DD HH:MM` format) from the user and calculates the number of days, hours, and minutes left until the meeting.

import datetime
```

```
def meeting_scheduler(meeting_datetime):
  # Parse the meeting date and time
  meeting_datetime = datetime.datetime.strptime(meeting_datetime, '%Y-%m-%d %H:%M')

  # Get the current date and time
  current_datetime = datetime.datetime.now()

  # Calculate the difference between the meeting date and time and the current date and time
  delta = meeting_datetime - current_datetime

  # Calculate the number of days, hours, and minutes left until the meeting
  days = delta.days
  hours = delta.seconds // 3600
  minutes = (delta.seconds // 60) % 60

  # Print the number of days, hours, and minutes left until the meeting
  print(f"There are {days} days, {hours} hours, and {minutes} minutes left until the meeting.")

# Get the meeting date and time from the user
meeting_datetime = input("Enter the meeting date and time (YYYY-MM-DD HH:MM): ")

# Schedule the meeting
meeting_scheduler(meeting_datetime)
```

```
Enter the meeting date and time (YYYY-MM-DD HH:MM): 2024-12-12  2:00
There are 176 days, 21 hours, and 8 minutes left until the meeting.
```

1. **Time Zone Converter**
   - Write a program that converts the current local time to UTC (Coordinated Universal Time). Display both the local time and the converted UTC time.

```
# prompt: 7. **Time Zone Converter**
#    - Write a program that converts the current local time to UTC (Coordinated Universal Time). Display both the local time and the converted UTC time.

import datetime

# Get the current local time
local_time = datetime.datetime.now()

# Convert the local time to UTC
utc_time = local_time.astimezone(datetime.timezone.utc)
```

```
# Print the local time and the converted UTC time
print(f"Local time: {local_time}")
print(f"UTC time: {utc_time}")

Local time: 2024-06-18 04:50:45.896331
UTC time: 2024-06-18 04:50:45.896331+00:00
```

1. **Day of the Week Calculator**
   - Write a script that asks the user to input a date (in `YYYY-MM-DD` format) and prints the day of the week for that date.

```
# prompt: 8. **Day of the Week Calculator**
#    - Write a script that asks the user to input a date (in `YYYY-MM-
DD` format) and prints the day of the week for that date.

def get_day_of_week(date):
  # Parse the date
  date = datetime.datetime.strptime(date, '%Y-%m-%d')

  # Get the day of the week
  day_of_week = date.strftime("%A")

  # Print the day of the week
  print(f"Day of the week: {day_of_week}")

# Get the date from the user
date = input("Enter a date (YYYY-MM-DD): ")

# Get the day of the week
get_day_of_week(date)

Enter a date (YYYY-MM-DD): 2024-11-12
Day of the week: Tuesday
```

1. **Recurring Event Notifier**
   - Develop a program that calculates the next 5 occurrences of a recurring event given a start date and a recurrence interval in days. Display the dates of these occurrences.

```
# prompt: 9. **Recurring Event Notifier**
#    - Develop a program that calculates the next 5 occurrences of a
recurring event given a start date and a recurrence interval in days.
Display the dates of these occurrences.

import datetime

def recurring_event_notifier(start_date, interval):
  # Parse the start date
  start_date = datetime.datetime.strptime(start_date, '%Y-%m-%d')
```

```
  # Initialize a list to store the next 5 occurrences
  occurrences = []

  # Calculate the next 5 occurrences
  for i in range(5):
    next_occurrence = start_date + datetime.timedelta(days=i *
interval)
    occurrences.append(next_occurrence)

  # Print the dates of the next 5 occurrences
  for occurrence in occurrences:
    print(occurrence.strftime('%Y-%m-%d'))

# Get the start date and recurrence interval from the user
start_date = input("Enter the start date (YYYY-MM-DD): ")
interval = int(input("Enter the recurrence interval in days: "))

# Notify the user about the next 5 occurrences
recurring_event_notifier(start_date, interval)
```

```
Enter the start date (YYYY-MM-DD): 2024-7-11
Enter the recurrence interval in days: 4
2024-07-11
2024-07-15
2024-07-19
2024-07-23
2024-07-27
```

1. **Time Since Event**
   - Create a program that asks the user to input a past date and time (in `YYYY-MM-DD HH:MM:SS` format) and calculates how much time has passed since that event. Display the result in years, months, days, hours, minutes, and seconds.

```
# prompt: 10. **Time Since Event**
#     - Create a program that asks the user to input a past date and
time (in `YYYY-MM-DD HH:MM:SS` format) and calculates how much time
has passed since that event. Display the result in years, months,
days, hours, minutes, and seconds.

import datetime

def time_since_event(past_datetime):
  # Parse the past date and time
  past_datetime = datetime.datetime.strptime(past_datetime, '%Y-%m-%d
%H:%M:%S')

  # Get the current date and time
  current_datetime = datetime.datetime.now()

  # Calculate the difference between the current date and time and the
```

```
past date and time
  delta = current_datetime - past_datetime

  # Extract the years, months, days, hours, minutes, and seconds
  years = delta.days // 365
  months = (delta.days % 365) // 30
  days = (delta.days % 365) % 30
  hours = delta.seconds // 3600
  minutes = (delta.seconds // 60) % 60
  seconds = delta.seconds % 60

  # Print the time since the event
  print(f"Time since the event: {years} years, {months} months, {days}
days, {hours} hours, {minutes} minutes, and {seconds} seconds.")

# Get the past date and time from the user
past_datetime = input("Enter the past date and time (YYYY-MM-DD
HH:MM:SS): ")

# Calculate and print the time since the event
time_since_event(past_datetime)
```

1. **Time Difference Between Cities**
   - Write a program that calculates the time difference between two cities given their respective time zones. Display the current time in both cities and the difference in hours and minutes.

```
# prompt: 11. **Time Difference Between Cities**
#     - Write a program that calculates the time difference between
two cities given their respective time zones. Display the current time
in both cities and the difference in hours and minutes.

import datetime
import pytz

def time_difference(city1, city2):
  # Get the current time in each city
  current_time_city1 = datetime.datetime.now(pytz.timezone(city1))
  current_time_city2 = datetime.datetime.now(pytz.timezone(city2))

  # Calculate the time difference
  time_difference = current_time_city1 - current_time_city2

  # Extract the hours and minutes from the time difference
  hours = time_difference.seconds // 3600
  minutes = (time_difference.seconds // 60) % 60

  # Print the current time in both cities and the time difference
  print(f"Current time in {city1}: {current_time_city1}")
  print(f"Current time in {city2}: {current_time_city2}")
```

```
  print(f"Time difference: {hours} hours and {minutes} minutes")

# Get the names of the two cities from the user
city1 = input("Enter the first city: ")
city2 = input("Enter the second city: ")

# Calculate and print the time difference
time_difference(city1, city2)
```

- Create a script that takes a file path as input and prints the last modification date and time of the file. If the file does not exist, display an appropriate message.

```
# prompt:     - Create a script that takes a file path as input and prints the last modification date and time of the file. If the file does not exist, display an appropriate message.

import os

def get_file_modification_date(file_path):
  # Check if the file exists
  if os.path.exists(file_path):
    # Get the last modification time of the file
    modification_time = os.path.getmtime(file_path)
    print( modification_time )

    # Convert the modification time to a datetime object
    modification_datetime =
datetime.datetime.fromtimestamp(modification_time)

    # Print the last modification date and time
    print(f"Last modification date and time: {modification_datetime}")
  else:
    # Print an error message if the file does not exist
    print(f"File not found: {file_path}")

# Get the file path from the user
file_path = input("Enter the file path: ")

# Get the last modification date and time of the file
get_file_modification_date(file_path)
```

```
Enter the file path: /content/report_2024-06-18.txt
1718706649.2354798
Last modification date and time: 2024-06-18 10:30:49.235480
```

1. **Work Shift Scheduler**
   - Develop a program that asks the user to input the start date and time of their work shift and the duration of the shift in hours. Calculate and display the end date and time of the shift.

```python
# prompt: 13. **Work Shift Scheduler**
#     - Develop a program that asks the user to input the start date and time of their work shift and the duration of the shift in hours. Calculate and display the end date and time of the shift.

import datetime

def work_shift_scheduler(start_datetime, duration):
  # Calculate the end date and time of the shift
  end_datetime = start_datetime + datetime.timedelta(hours=duration)

  # Print the start and end date and time of the shift
  print(f"Start date and time: {start_datetime}")
  print(f"End date and time: {end_datetime}")

# Get the start date and time of the shift from the user
start_date = input("Enter the start date (YYYY-MM-DD): ")
start_time = input("Enter the start time (HH:MM): ")

# Combine the start date and time into a datetime object
start_datetime = datetime.datetime.strptime(f"{start_date} {start_time}", '%Y-%m-%d %H:%M')

# Get the duration of the shift from the user
duration = int(input("Enter the duration of the shift in hours: "))

# Schedule the work shift
work_shift_scheduler(start_datetime, duration)
```

```
Enter the start date (YYYY-MM-DD): 2024-12-12
Enter the start time (HH:MM): 3:00
Enter the duration of the shift in hours: 5
Start date and time: 2024-12-12 03:00:00
End date and time: 2024-12-12 08:00:00
```

1. **Birthdays in a Month**
   - Write a program that accepts a list of names and their birthdates. Then, ask the user to input a month (as a number). Display the names of people whose birthdays are in that month.

```python
# prompt: 14. **Birthdays in a Month**
#     - Write a program that accepts a list of names and their birthdates. Then, ask the user to input a month (as a number). Display the names of people whose birthdays are in that month.

birthdays = {
    "Alice": "January 15",
    "Bob": "February 23",
    "Carol": "March 10",
    "Dave": "April 5",
```

```
    "Eve": "May 20",
    "Frank": "June 12",
    "Grace": "July 8",
    "Harry": "August 19",
    "Ivy": "September 25",
    "Jack": "October 3",
    "Kelly": "November 17",
    "Lisa": "December 28"
}

month = int(input("Enter a month (1-12): "))

for name, birthday in birthdays.items():
    month_str = birthday.split()[1]
    month_num = int(month_str)
    if month_num == month:
        print(name)
```

```
Enter a month (1-12): 12
Frank
```

1. **Future Date Calculator**
   - Create a program that asks the user to input a number of days, weeks, or months and calculates the future date from today based on the input.

```
# prompt: 15. **Future Date Calculator**
#     - Create a program that asks the user to input a number of days,
weeks, or months and calculates the future date from today based on
the input.

def future_date_calculator(num, unit):
  # Get the current date
  current_date = datetime.date.today()

  # Calculate the future date based on the input
  if unit == "days":
    future_date = current_date + datetime.timedelta(days=num)
  elif unit == "weeks":
    future_date = current_date + datetime.timedelta(weeks=num)
  elif unit == "months":
    future_date = current_date + datetime.timedelta(days=num * 30)
  else:
    print("Invalid unit. Please enter 'days', 'weeks', or 'months'.")
    return

  # Print the future date
  print(f"The future date is: {future_date}")

# Get the number and unit from the user
num = int(input("Enter the number of days, weeks, or months: "))
```

```
unit = input("Enter the unit (days, weeks, or months): ").lower()

# Calculate and print the future date
future_date_calculator(num, unit)
```

```
Enter the number of days, weeks, or months: 7
Enter the unit (days, weeks, or months): days
The future date is: 2024-06-25
```

Practice Questions on math module:

1. Write a program that asks the user to input the radius of a circle. Calculate and print the area and circumference of the circle using the value of π from the math module.

```
# prompt: 1. Write a program that asks the user to input the radius of a circle. Calculate and print the area and circumference of the circle using the value of π from the math module.

import math

# Get the radius of the circle from the user
radius = float(input("Enter the radius of the circle: "))

# Calculate the area of the circle
area = math.pi * radius ** 2

# Calculate the circumference of the circle
circumference = 2 * math.pi * radius

# Print the area and circumference of the circle
print(f"Area of the circle: {area}")
print(f"Circumference of the circle: {circumference}")
```

```
Enter the radius of the circle: 3
Area of the circle: 28.274333882308138
Circumference of the circle: 18.84955592153876
```

1. Create a program that accepts the lengths of the two shorter sides of a right triangle. Calculate and print the length of the hypotenuse using the Pythagorean theorem.

```
# prompt: 2. Create a program that accepts the lengths of the two shorter sides of a right triangle. Calculate and print the length of the hypotenuse using the Pythagorean theorem.

import math

def calculate_hypotenuse(a, b):
  """
  Calculates the length of the hypotenuse of a right triangle given the lengths of the two shorter sides.
```

```
  Args:
    a: The length of one of the shorter sides.
    b: The length of the other shorter side.

  Returns:
    The length of the hypotenuse.
  """

  # Use the Pythagorean theorem to calculate the length of the
hypotenuse
  hypotenuse = math.sqrt(a ** 2 + b ** 2)

  # Return the length of the hypotenuse
  return hypotenuse

# Get the lengths of the two shorter sides from the user
a = float(input("Enter the length of one of the shorter sides: "))
b = float(input("Enter the length of the other shorter side: "))

# Calculate and print the length of the hypotenuse
hypotenuse = calculate_hypotenuse(a, b)
print(f"The length of the hypotenuse is: {hypotenuse}")
```

```
Enter the length of one of the shorter sides: 12
Enter the length of the other shorter side: 12
The length of the hypotenuse is: 16.97056274847714
```

1. Develop a program to calculate compound interest. Ask the user for the principal amount, the annual interest rate, the number of times interest is compounded per year, and the number of years. Use the math module to perform the calculations.

```
# prompt: 3. Develop a program to calculate compound interest. Ask the
user for the principal amount, the annual interest rate, the number of
times interest is compounded per year, and the number of years. Use
the math module to perform the calculations.

def compound_interest(principal, rate, times_compounded, years):
  """
  Calculates the compound interest earned on a given principal amount.

  Args:
    principal: The principal amount.
    rate: The annual interest rate.
    times_compounded: The number of times interest is compounded per
year.
    years: The number of years.

  Returns:
    The amount of compound interest earned.
  """
```

```
  # Calculate the annual interest rate in decimal form
  annual_rate = rate / 100

  # Calculate the number of times interest is compounded in total
  total_compounding = times_compounded * years

  # Use the compound interest formula to calculate the amount of
interest earned
  interest = principal * (1 + annual_rate / times_compounded) **
total_compounding - principal

  # Return the amount of interest earned
  return interest

# Get the principal amount, annual interest rate, number of times
interest is compounded per year, and number of years from the user
principal = float(input("Enter the principal amount: "))
rate = float(input("Enter the annual interest rate: "))
times_compounded = int(input("Enter the number of times interest is
compounded per year: "))
years = int(input("Enter the number of years: "))

# Calculate and print the amount of compound interest earned
interest = compound_interest(principal, rate, times_compounded, years)
print(f"The amount of compound interest earned is: {interest}")
```

```
Enter the principal amount: 1000
Enter the annual interest rate: 2
Enter the number of times interest is compounded per year: 2
Enter the number of years: 5
The amount of compound interest earned is: 104.62212541120448
```

1. Write a script that solves a quadratic equation of the form (ax^2 + bx + c = 0). Ask the user to input the coefficients a, b, and c. Calculate and print the solutions using the quadratic formula.

```
# prompt: 4. Write a script that solves a quadratic equation of the
form \(ax^2 + bx + c = 0\). Ask the user to input the coefficients a,
b, and c. Calculate and print the solutions using the quadratic
formula.

import math

# Get the coefficients a, b, and c from the user
a = float(input("Enter the coefficient a: "))
b = float(input("Enter the coefficient b: "))
c = float(input("Enter the coefficient c: "))

# Calculate the discriminant
```

```
discriminant = b ** 2 - 4 * a * c

# Check if the discriminant is negative
if discriminant < 0:
  print("The equation has no real solutions.")
elif discriminant == 0:
  # Calculate the single solution
  solution = -b / (2 * a)
  print(f"The equation has one solution: {solution}")
else:
  # Calculate the two solutions
  solution1 = (-b - math.sqrt(discriminant)) / (2 * a)
  solution2 = (-b + math.sqrt(discriminant)) / (2 * a)
  print(f"The equation has two solutions: {solution1} and
{solution2}")
```

```
Enter the coefficient a: 2
Enter the coefficient b: 3
Enter the coefficient c: 5
The equation has no real solutions.
```

1. Create a program that asks the user to input a number and a base. Calculate and print the logarithm of the number to the given base using the math module.

```
# prompt: 5. Create a program that asks the user to input a number and
a base. Calculate and print the logarithm of the number to the given
base using the math module.

import math

# Get the number and base from the user
number = float(input("Enter a number: "))
base = float(input("Enter the base: "))

# Calculate the logarithm of the number to the given base
logarithm = math.log(number, base)

# Print the logarithm
print(f"The logarithm of {number} to the base {base} is: {logarithm}")
```

```
Enter a number: 10
Enter the base: 10
The logarithm of 10.0 to the base 10.0 is: 1.0
```

1. Develop a script that models exponential growth. Ask the user for the initial amount, the growth rate, and the number of periods. Calculate and print the final amount using the exponential function from the math module.

```
# prompt: 6. Develop a script that models exponential growth. Ask the
user for the initial amount, the growth rate, and the number of
periods. Calculate and print the final amount using the exponential
```

```
function from the math module.

# Get the initial amount, growth rate, and number of periods from the
user
initial_amount = float(input("Enter the initial amount: "))
growth_rate = float(input("Enter the growth rate: "))
num_periods = int(input("Enter the number of periods: "))

# Calculate the final amount using the exponential function
final_amount = initial_amount * (1 + growth_rate) ** num_periods

# Print the final amount
print(f"The final amount is: {final_amount}")
```

```
Enter the initial amount: 10
Enter the growth rate: 2
Enter the number of periods: 3
The final amount is: 270.0
```

1. Write a program that accepts an angle in degrees from the user. Convert the angle to radians and calculate the sine, cosine, and tangent of the angle. Print the results.

```
# prompt: 7. Write a program that accepts an angle in degrees from the
user. Convert the angle to radians and calculate the sine, cosine, and
tangent of the angle. Print the results.

import math

# Get the angle in degrees from the user
angle_deg = float(input("Enter the angle in degrees: "))

# Convert the angle to radians
angle_rad = math.radians(angle_deg)

# Calculate the sine, cosine, and tangent of the angle
sine = math.sin(angle_rad)
cosine = math.cos(angle_rad)
tangent = math.tan(angle_rad)

# Print the results
print(f"Sine: {sine}")
print(f"Cosine: {cosine}")
print(f"Tangent: {tangent}")
```

```
Enter the angle in degrees: 12
Sine: 0.20791169081775934
Cosine: 0.9781476007338057
Tangent: 0.21255656167002213
```

1. Create a script that calculates the distance between two points in a 2D plane. Ask the user for the coordinates of the points (x1, y1) and (x2, y2). Use the distance formula to calculate and print the distance.

```
# prompt: 8. Create a script that calculates the distance between two points in a 2D plane. Ask the user for the coordinates of the points (x1, y1) and (x2, y2). Use the distance formula to calculate and print the distance.

import math

# Get the coordinates of the points from the user
x1 = float(input("Enter the x-coordinate of the first point: "))
y1 = float(input("Enter the y-coordinate of the first point: "))
x2 = float(input("Enter the x-coordinate of the second point: "))
y2 = float(input("Enter the y-coordinate of the second point: "))

# Calculate the distance between the points using the distance formula
distance = math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2)

# Print the distance
print(f"The distance between the two points is: {distance}")
```

```
Enter the x-coordinate of the first point: 2
Enter the y-coordinate of the first point: 3
Enter the x-coordinate of the second point: 1
Enter the y-coordinate of the second point: 5
The distance between the two points is: 2.23606797749979
```

1. Write a program that asks the user to input a non-negative integer. Calculate and print the factorial of the number using the math module.

```
# prompt: 9. Write a program that asks the user to input a non-negative integer. Calculate and print the factorial of the number using the math module.

# Get a non-negative integer from the user
number = int(input("Enter a non-negative integer: "))

# Calculate the factorial of the number using the math module
factorial = math.factorial(number)

# Print the factorial
print(f"The factorial of {number} is: {factorial}")
```

```
Enter a non-negative integer: 5
The factorial of 5 is: 120
```

1. Develop a program that converts an angle given in degrees to radians and vice versa. Ask the user for the angle and the conversion direction. Perform and print the conversion.

```python
# prompt: 10. Develop a program that converts an angle given in
degrees to radians and vice versa. Ask the user for the angle and the
conversion direction. Perform and print the conversion.

import math

def convert_angle(angle, direction):
  """
  Converts an angle between degrees and radians.

  Args:
    angle: The angle to convert.
    direction: The conversion direction. Either "degrees_to_radians"
or "radians_to_degrees".

  Returns:
    The converted angle.
  """

  if direction == "degrees_to_radians":
    return math.radians(angle)
  elif direction == "radians_to_degrees":
    return math.degrees(angle)
  else:
    raise ValueError("Invalid direction. Please enter
'degrees_to_radians' or 'radians_to_degrees'.")

# Get the angle and conversion direction from the user
angle = float(input("Enter the angle: "))
direction = input("Enter the conversion direction (degrees_to_radians
or radians_to_degrees): ").lower()

# Convert and print the angle
converted_angle = convert_angle(angle, direction)
print(f"The converted angle is: {converted_angle}")
```

```
Enter the angle: 12
Enter the conversion direction (degrees_to_radians or
radians_to_degrees): degrees_to_radians
The converted angle is: 0.20943951023931956
```