Day 11 of Training at Ansh Info Tech

Topics Covered

- **Math Module in Python**
  - Trignometric, Logarithmic, Exponential etc. Functions
- **DateTime Module in Python**
  - today(), now() etc. Functions
- **Numpy Introduction in Python**
  - Real Life Applications
  - Numpy Arrays
  - Operations like Reshape, indexing
  - N-dimensional Arrays
  - Mathematical Functions
  - Random Number Generation
- **25 Questions on Math Module**
- **25 Questions on DateTime Module**
- **Practice Worksheet on Numpy Module**

```
# prompt: Math Module Important Functions

import math

# Calculate the square root of a number
square_root = math.sqrt(16)
print(f"The square root of 16 is: {square_root}")

# Calculate the power of a number
power = math.pow(2, 3)
print(f"2 raised to the power of 3 is: {power}")

# Calculate the factorial of a number
factorial = math.factorial(5)
print(f"The factorial of 5 is: {factorial}")

# Calculate the sine of an angle in radians
sine = math.sin(math.radians(30))
print(f"The sine of 30 degrees is: {sine}")

# Calculate the cosine of an angle in radians
cosine = math.cos(math.radians(45))
print(f"The cosine of 45 degrees is: {cosine}")

# Calculate the tangent of an angle in radians
tangent = math.tan(math.radians(60))
print(f"The tangent of 60 degrees is: {tangent}")
```

```
# prompt: DateTime Functions in Python

import datetime

# Get the current date and time
now = datetime.datetime.now()
print("Current date and time:", now)

# Get the current date
today = datetime.date.today()
print("Today's date:", today)

# Get the current time
current_time = datetime.datetime.now().time()
print("Current time:", current_time)

# Get the year, month, and day
year = now.year
month = now.month
day = now.day

print("Year:", year)
print("Month:", month)
print("Day:", day)

# Get the weekday number (0 for Monday, 1 for Tuesday, etc.)
weekday_number = now.weekday()
print("Weekday number:", weekday_number)

# Get the weekday name
weekday_name = datetime.date.today().strftime("%A")
print("Weekday name:", weekday_name)

# Get the ISO 8601 week number
iso_week_number = now.isocalendar()[1]
print("ISO 8601 week number:", iso_week_number)
```

```
Current date and time: 2024-06-18 04:05:40.629756
Today's date: 2024-06-18
Current time: 04:05:40.630680
Year: 2024
Month: 6
Day: 18
Weekday number: 1
Weekday name: Tuesday
ISO 8601 week number: 25
```

Introduction to the Math Module

The math module in Python provides access to mathematical functions that perform mathematical operations on numerical data. It includes functions for basic arithmetic, trigonometry, logarithms, exponentiation, and more complex operations. Understanding and utilizing the math module is essential for performing advanced mathematical computations in Python.

Key Functions of the Math Module

1.Basic Arithmetic Operations

- math.sqrt(x): Returns the square root of x.
- math.pow(x, y): Returns x raised to the power of y.
- math.factorial(x): Returns the factorial of x.

2.Trigonometric Functions

- math.sin(x), math.cos(x), math.tan(x): Returns the sine, cosine, and tangent of x (in radians).
- math.radians(x), math.degrees(x): Convert angles from degrees to radians and vice versa.

3.Logarithmic and Exponential Functions

- math.log(x, base): Returns the logarithm of x to the given base (default is natural logarithm).
- math.exp(x): Returns e raised to the power of x.

Constants

- math.pi: Mathematical constant π (pi).
- math.e: Mathematical constant e (base of natural logarithm).

Basic Arithmetic Operations

```
import math

# Calculate square root of a number
print(math.sqrt(25))

# Calculate 2 raised to the power of 3
print(math.pow(2, 3))

# Calculate factorial of 5
print(math.factorial(5))
```

```
5.0
8.0
120
```

Trigonometric Functions

```
import math

# Calculate sine of 45 degrees (converted to radians)
print(math.sin(math.radians(45)))

# Calculate cosine of 30 degrees (converted to radians)
print(math.cos(math.radians(30)))

# Calculate tangent of 60 degrees (converted to radians)
print(math.tan(math.radians(60)))
```

```
0.7071067811865475
0.8660254037844387
1.7320508075688767
```

Logarithmic and Exponential Functions

```
import math

# Calculate natural logarithm of 10
print(math.log(10))

# Calculate logarithm of 100 to base 10
print(math.log(100, 10))

# Calculate e raised to the power of 2
print(math.exp(2))
```

```
2.302585092994046
2.0
7.38905609893065
```

```
print(math.pi)
```

```
3.141592653589793
```

```
print(math.e)
```

```
2.718281828459045
```

Practical Applications

- Scientific Computing: Use math for scientific calculations, numerical analysis, and modeling.
- Engineering Applications: Perform calculations related to physics, geometry, and signal processing.
- Financial Calculations: Compute interest rates, present values, and other financial metrics.
- Statistical Analysis: Utilize mathematical functions for statistical operations such as probability distributions and hypothesis testing.

Introduction to the Datetime Module

The `datetime` module in Python provides classes for manipulating dates and times. It offers functionality to work with dates, times, timedeltas (differences between dates or times), time zones, and formatting options. Understanding and utilizing the `datetime` module is crucial for handling date and time data effectively in Python programming.

Key Classes and Functions of the Datetime Module

1. **Date Objects**
   - `datetime.date(year, month, day)` : Represents a date (year, month, day).
   - `date.today()` : Returns the current local date.
   - `date.strftime(format)` : Formats a date object into a string using specified format codes.
2. **Time Objects**
   - `datetime.time(hour, minute, second, microsecond)` : Represents a time (hour, minute, second, microsecond).
   - `time.strftime(format)` : Formats a time object into a string using specified format codes.
3. **Datetime Objects**
   - `datetime.datetime(year, month, day, hour, minute, second, microsecond)` : Represents a datetime (date and time).
   - `datetime.now()` : Returns the current local datetime.
   - `datetime.strptime(date_string, format)` : Parses a string into a datetime object based on the format.
4. **Timedelta Objects**
   - `datetime.timedelta(days, seconds, microseconds, milliseconds, minutes, hours, weeks)` : Represents a duration or difference between two dates or times.

Practical Applications

- **Date and Time Calculations**: Perform calculations such as date differences, time durations, and scheduling.
- **Data Analysis**: Manipulate timestamps and analyze time series data.
- **Web Applications**: Manage session timeouts, scheduling tasks, or displaying timestamps.
- **Database Operations**: Store and retrieve dates/times from databases and handle timezone conversions.

```
#### Examples and Demonstrations

##### Working with Date Objects


import datetime

# Create a date object
d = datetime.date(2024, 6, 17)
print(d)

# Get today's date
today = datetime.date.today()
print(today)

# Format date as string
print(today.strftime('%Y-%m-%d'))

print(today.strftime('%d-%m-%Y'))
```

```
2024-06-17
2024-06-18
2024-06-18
18-06-2024
```

```
## Working with Time Objects
```

```
import datetime

# Create a time object
t = datetime.time(10, 30, 0)
print(t)

# Format time as string
print(t.strftime('%H:%M:%S'))
```

```
10:30:00
10:30:00
```

```
# Working with Datetime Objects
```

```
import datetime

# Create a datetime object
dt = datetime.datetime(2024, 6, 17, 10, 30, 0)
print(dt)

# Get current datetime
now = datetime.datetime.now()
print(now)

# Parse string to datetime
dt_str = '2024-06-17 10:30:00'
parsed_dt = datetime.datetime.strptime(dt_str, '%Y-%m-%d %H:%M:%S')
print(parsed_dt)
```

```
2024-06-17 10:30:00
2024-06-18 04:01:28.375770
2024-06-17 10:30:00
```

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.
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 between these two times in hours, minutes, and seconds.
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.
4. *Weekly Report Generator*
    - Write a script that generates a weekly report file named report_.txt, where is the current date. The file should contain a timestamp of when the report was generated.
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.
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.
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.

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.
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.
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.
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.
12. *File Modification Tracker*
    - 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.
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.
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.
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.
16. *Leap Year Checker*
    - Write a program that asks the user to input a year and checks whether it is a leap year or not. Display an appropriate message.
17. *Appointment Reminder*
    - Develop a program that accepts the date and time of an appointment (in YYYY-MM-DD HH:MM:SS format) and calculates the remaining time until the appointment. Display a reminder message if the appointment is within 24 hours.
18. *Monthly Calendar Generator*
    - Write a script that generates and prints the calendar for a specific month and year input by the user.
19. *Vacation Countdown*
    - Create a program that asks the user to input the start date of their vacation and calculates the number of days left until the vacation starts.
20. *Elapsed Time Calculator*
    - Write a program that measures the time taken to execute a block of code. Use the datetime module to capture the start and end times and print the elapsed time in seconds.
21. *Anniversary Reminder*
    - Develop a program that takes a list of anniversary dates and checks which anniversaries are coming up within the next 30 days. Display the names and dates of these anniversaries.
22. *Work Hours Tracker*
    - Create a script that allows the user to input their clock-in and clock-out times for each day of the week. Calculate and display the total number of hours worked in the week.
23. *Date Difference Calculator*
    - Write a program that asks the user to input two dates (in YYYY-MM-DD format) and calculates the difference between them in days, weeks, and months.
24. *Daily Task Scheduler*

- Develop a daily task scheduler that allows the user to input tasks with specific times. Display a schedule of tasks for the day and notify the user of upcoming tasks.

25. *Day Counter*
    - Create a program that counts the number of days from the current date to a user-specified future date. Display the result.

```
from datetime import datetime, timedelta
import calendar
import pytz
import os
import time

# 1. Event Reminder Application
def event_reminder(event_name, event_date):
    today = datetime.today()
    event_date = datetime.strptime(event_date, "%Y-%m-%d")
    days_left = (event_date - today).days
    return f"Event: {event_name} is in {days_left} days."

# 2. Time Duration Calculation
def time_duration(start_time, end_time):
    start_time = datetime.strptime(start_time, "%H:%M:%S")
    end_time = datetime.strptime(end_time, "%H:%M:%S")
    duration = end_time - start_time
    return f"Duration: {duration}"

# 3. Date of Birth and Age Calculation
def calculate_age(dob):
    today = datetime.today()
    dob = datetime.strptime(dob, "%Y-%m-%d")
    age = today - dob
    years = age.days // 365
    months = (age.days % 365) // 30
    days = (age.days % 365) % 30
    return f"Age: {years} years, {months} months, {days} days"

# 4. Weekly Report Generator
def weekly_report():
    today = datetime.today().strftime('%Y-%m-%d')
    with open(f"report_{today}.txt", "w") as file:
        file.write(f"Weekly report generated on {today}\n")
    return f"Report generated: report_{today}.txt"

# 5. Project Deadline Tracker
def project_deadline(project_name, deadline_date):
    today = datetime.today()
    deadline_date = datetime.strptime(deadline_date, "%Y-%m-%d")
    days_left = (deadline_date - today).days
    if days_left <= 7:
        return f"Project: {project_name} deadline is in {days_left} days."
    else:
        return f"Project: {project_name} deadline is not within 7 days."

# 6. Meeting Scheduler
def meeting_scheduler(meeting_datetime):
    today = datetime.now()
    meeting_datetime = datetime.strptime(meeting_datetime, "%Y-%m-%d %H:%M")
    delta = meeting_datetime - today
    days = delta.days
    hours, remainder = divmod(delta.seconds, 3600)
    minutes, _ = divmod(remainder, 60)
    return f"Time until meeting: {days} days, {hours} hours, and {minutes} minutes."

# 7. Time Zone Converter
def timezone_converter():
    local_time = datetime.now()
    utc_time = datetime.now(pytz.utc)
    return f"Local time: {local_time}, UTC time: {utc_time}"

# 8. Day of the Week Calculator
def day_of_week(date):
    date = datetime.strptime(date, "%Y-%m-%d")
    return f"Day of the week: {date.strftime('%A')}"

# 9. Recurring Event Notifier
def recurring_event(start_date, interval_days):
    start_date = datetime.strptime(start_date, "%Y-%m-%d")
    occurrences = [start_date + timedelta(days=interval_days * i) for i in range(5)]
    return f"Next 5 occurrences: {[date.strftime('%Y-%m-%d') for date in occurrences]}"

# 10. Time Since Event
def time_since_event(event_datetime):
    event_datetime = datetime.strptime(event_datetime, "%Y-%m-%d %H:%M:%S")
```

```
    now = datetime.now()
    delta = now - event_datetime
    years = delta.days // 365
    months = (delta.days % 365) // 30
    days = (delta.days % 365) % 30
    hours, remainder = divmod(delta.seconds, 3600)
    minutes, seconds = divmod(remainder, 60)
    return f"Time since event: {years} years, {months} months, {days} days, {hours} hours, {minutes} minutes, {seconds} seconds"

# 11. Time Difference Between Cities
def time_difference_between_cities(city1_tz, city2_tz):
    now = datetime.now(pytz.utc)
    city1_time = now.astimezone(pytz.timezone(city1_tz))
    city2_time = now.astimezone(pytz.timezone(city2_tz))
    difference = (city1_time.utcoffset() - city2_time.utcoffset()).total_seconds() / 3600
    return f"City1 time: {city1_time}, City2 time: {city2_time}, Difference: {difference} hours"

# 12. File Modification Tracker
def file_modification_tracker(file_path):
    try:
        modification_time = datetime.fromtimestamp(os.path.getmtime(file_path))
        return f"Last modification date and time: {modification_time}"
    except FileNotFoundError:
        return "File does not exist."

# 13. Work Shift Scheduler
def work_shift_scheduler(start_datetime, shift_duration_hours):
    start_datetime = datetime.strptime(start_datetime, "%Y-%m-%d %H:%M:%S")
    end_datetime = start_datetime + timedelta(hours=shift_duration_hours)
    return f"Shift ends at: {end_datetime}"

# 14. Birthdays in a Month
def birthdays_in_month(birthdays, month):
    birthdays_in_month = [name for name, date in birthdays.items() if datetime.strptime(date, "%Y-%m-%d").month == month]
    return f"Birthdays in month {month}: {birthdays_in_month}"

# 15. Future Date Calculator
def future_date_calculator(days=0, weeks=0, months=0):
    today = datetime.today()
    future_date = today + timedelta(days=days) + timedelta(weeks=weeks) + timedelta(days=months*30)
    return f"Future date: {future_date.strftime('%Y-%m-%d')}"

# 16. Leap Year Checker
def leap_year_checker(year):
    if (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0):
        return f"{year} is a leap year."
    else:
        return f"{year} is not a leap year."

# 17. Appointment Reminder
def appointment_reminder(appointment_datetime):
    appointment_datetime = datetime.strptime(appointment_datetime, "%Y-%m-%d %H:%M:%S")
    now = datetime.now()
    delta = appointment_datetime - now
    if delta.days < 1:
        return f"Appointment is within 24 hours!"
    else:
        return f"Appointment is in {delta.days} days."

# 18. Monthly Calendar Generator
def monthly_calendar_generator(year, month):
    return calendar.month(year, month)

# 19. Vacation Countdown
def vacation_countdown(vacation_start_date):
    vacation_start_date = datetime.strptime(vacation_start_date, "%Y-%m-%d")
    days_left = (vacation_start_date - datetime.today()).days
    return f"Days until vacation: {days_left}"

# 20. Elapsed Time Calculator
def elapsed_time_calculator():
    start = datetime.now()
    # Simulate code execution with sleep
    time.sleep(1)
    end = datetime.now()
    elapsed = (end - start).total_seconds()
    return f"Elapsed time: {elapsed} seconds"
```

```
# 21. Anniversary Reminder
def anniversary_reminder(anniversaries):
    today = datetime.today()
    upcoming_anniversaries = [(name, date) for name, date in anniversaries.items() if 0 <= (datetime.strptime(date, "%Y-%m-%d") - today).day
    return f"Upcoming anniversaries: {upcoming_anniversaries}"

# 22. Work Hours Tracker
def work_hours_tracker(work_hours):
    total_hours = sum([(datetime.strptime(end, "%H:%M:%S") - datetime.strptime(start, "%H:%M:%S")).total_seconds() / 3600 for start, end in
    return f"Total hours worked in the week: {total_hours}"

# 23. Date Difference Calculator
def date_difference_calculator(date1, date2):
    date1 = datetime.strptime(date1, "%Y-%m-%d")
    date2 = datetime.strptime(date2, "%Y-%m-%d")
    delta = abs(date2 - date1)
    weeks = delta.days // 7
    months = delta.days // 30
    return f"Difference: {delta.days} days, {weeks} weeks, {months} months"

# 24. Daily Task Scheduler
def daily_task_scheduler(tasks):
    now = datetime.now()
    upcoming_tasks = [task for task, time in tasks if datetime.strptime(time, "%H:%M:%S").time() > now.time()]
    return f"Upcoming tasks: {upcoming_tasks}"

# 25. Day Counter
def day_counter(future_date):
    future_date = datetime.strptime(future_date, "%Y-%m-%d")
    days_left = (future_date - datetime.today()).days
    return f"Days until {future_date.strftime('%Y-%m-%d')}: {days_left}"

# Example usage:

# 1. Event Reminder
print(event_reminder("Birthday", "2024-07-01"))

# 2. Time Duration Calculation
print(time_duration("12:30:00", "14:45:00"))

# 3. Date of Birth and Age Calculation
print(calculate_age("1990-06-15"))

# 4. Weekly Report Generator
print(weekly_report())

# 5. Project Deadline Tracker
print(project_deadline("Project Alpha", "2024-06-25"))

# 6. Meeting Scheduler
print(meeting_scheduler("2024-06-20 15:30"))

# 7. Time Zone Converter
print(timezone_converter())

# 8. Day of the Week Calculator
print(day_of_week("2024-06-18"))

# 9. Recurring Event Notifier
print(recurring_event("2024-06-01", 7))

# 10. Time Since Event
print(time_since_event("2023-06-15 10:00:00"))

# 11. Time Difference Between Cities
print(time_difference_between_cities("America/New_York", "Europe/London"))

# 12. File Modification Tracker
print(file_modification_tracker("example.txt"))

# 13. Work Shift Scheduler
print(work_shift_scheduler("2024-06-18 09:00:00", 8))

# 14. Birthdays in a Month
birthdays = {"Alice": "1990-06-15", "Bob": "1985-07-23", "Charlie": "1992-06-30"}
print(birthdays_in_month(birthdays, 6))
```

```
# 15. Future Date Calculator
print(future_date_calculator(days=10, weeks=2, months=1))

# 16. Leap Year Checker
print(leap_year_checker(2024))

# 17. Appointment Reminder
print(appointment_reminder("2024-06-19 15:00:00"))

# 18. Monthly Calendar Generator
print(monthly_calendar_generator(2024, 6))

# 19. Vacation Countdown
print(vacation_countdown("2024-07-01"))

# 20. Elapsed Time Calculator
print(elapsed_time_calculator())

# 21. Anniversary Reminder
anniversaries = {"Wedding": "2024-06-20", "Engagement": "2024-07-10"}
print(anniversary_reminder(anniversaries))

# 22. Work Hours Tracker
work_hours = [("09:00:00", "17:00:00"), ("09:00:00", "17:00:00"), ("09:00:00", "17:00:00"), ("09:00:00", "17:00:00"), ("09:00:00", "17:00:00")]
print(work_hours_tracker(work_hours))

# 23. Date Difference Calculator
print(date_difference_calculator("2024-06-01", "2024-06-18"))

# 24. Daily Task Scheduler
tasks = [("Meeting", "10:00:00"), ("Lunch", "12:00:00"), ("Call", "15:00:00")]
print(daily_task_scheduler(tasks))

# 25. Day Counter
print(day_counter("2024-07-01"))
```

```
Event: Birthday is in 12 days.
Duration: 2:15:00
Age: 34 years, 0 months, 12 days
Report generated: report_2024-06-18.txt
Project: Project Alpha deadline is in 6 days.
Time until meeting: 2 days, 11 hours, and 6 minutes.
Local time: 2024-06-18 04:23:40.807193, UTC time: 2024-06-18 04:23:40.807197+00:00
Day of the week: Tuesday
Next 5 occurrences: ['2024-06-01', '2024-06-08', '2024-06-15', '2024-06-22', '2024-06-29']
Time since event: 1 years, 0 months, 3 days, 18 hours, 23 minutes, 40 seconds
City1 time: 2024-06-18 00:23:40.809530-04:00, City2 time: 2024-06-18 05:23:40.809530+01:00, Difference: -5.0 hours
File does not exist.
Shift ends at: 2024-06-18 17:00:00
Birthdays in month 6: ['Alice', 'Charlie']
Future date: 2024-08-11
2024 is a leap year.
Appointment is in 1 days.
     June 2024
Mo Tu We Th Fr Sa Su
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30

Days until vacation: 12
Elapsed time: 1.000291 seconds
Upcoming anniversaries: [('Wedding', '2024-06-20'), ('Engagement', '2024-07-10')]
Total hours worked in the week: 40.0
Difference: 17 days, 2 weeks, 0 months
Upcoming tasks: ['Meeting', 'Lunch', 'Call']
Days until 2024-07-01: 12
```

Numpy Library

- N-dimensional Arrays
- Mathematical Functions
- Linear Algebra

- Random Number Generation

```
import numpy as np
random = np.random.randint(10)
random
```

```
8
```

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.
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.
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.
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.
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.
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.
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.
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.
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.
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.
11. Create a program that asks the user to input the radius of a sphere. Calculate and print the surface area and volume of the sphere using the value of π from the math module.
12. Write a script to calculate the Body Mass Index (BMI). Ask the user for their weight in kilograms and height in meters. Calculate and print the BMI using the formula (\text{BMI} = \frac{\text{weight}}{\text{height}^2}).
13. Develop a program that converts a temperature given in Celsius to Fahrenheit. Ask the user for the temperature in Celsius and print the converted temperature.
14. Write a script that calculates the distance between two points in a 3D space. Ask the user for the coordinates of the points (x1, y1, z1) and (x2, y2, z2). Use the 3D distance formula to calculate and print the distance.
15. Create a simple calculator program that performs addition, subtraction, multiplication, and division. Ask the user for two numbers and the operation to perform. Use the appropriate functions from the math module for the calculations.
16. Develop a program to calculate the harmonic mean of a list of numbers. Ask the user to input the numbers. Use the math module to calculate and print the harmonic mean.
17. Write a script to verify the trigonometric identity (\sin^2(x) + \cos^2(x) = 1). Ask the user for an angle in degrees, convert it to radians, and verify the identity using the math module.
18. Create a program to calculate the standard deviation of a list of numbers. Ask the user to input the numbers. Use the math module to perform the calculations and print the standard deviation.
19. Write a script that calculates the greatest common divisor (GCD) and least common multiple (LCM) of two numbers. Ask the user for the numbers and use the math module to calculate and print the GCD and LCM.
20. Develop a program that models logarithmic growth. Ask the user for the initial amount, the growth rate, and the number of periods. Calculate and print the final amount using the logarithmic function from the math module.
21. Create a script that generates and prints the values of a sine wave for a given range of angles. Ask the user for the start and end angles (in degrees) and the step size. Use the math module to calculate and print the sine values.

22. Write a program to check if a given number is prime. Ask the user to input a number and determine if it is a prime number using appropriate mathematical checks.
23. Develop a script to calculate the area of a triangle given its three sides using Heron's formula. Ask the user for the lengths of the sides and calculate the area using the math module.
24. Create a program to calculate the future value of an investment. Ask the user for the principal amount, annual interest rate, and number of years. Use the math module to calculate and print the future value.
25. Write a script that calculates the hyperbolic sine, cosine, and tangent of a given number. Ask the user for the number and print the results using the math module.

```
import math

# Question 1: Calculate Area and Circumference using Math.pi

radius = float(input("Enter the radius of the circle: "))
area = math.pi * radius ** 2
circumference = 2 * math.pi * radius
print("Area:", area)
print("Circumference:", circumference)

# Question 2: Find Hypotenuse of a right triangle using Pythagoras Theorem

perpendicular = float(input("Enter the Perpendicular Length: "))
base = float(input("Enter the Base Length: "))
hypotenuse = abs(math.sqrt(perpendicular ** 2 + base ** 2))
print("Hypotenuse:", hypotenuse)

# Question 3: Calculate Compound Interest

principal = float(input("Enter the Principal Amount: "))
rate = float(input("Enter the Annual Interest Rate: "))
periods = int(input("Enter the Number of Times Interest is Compounded per Year: "))
years = int(input("Enter the Number of Years: "))
amount = principal * (1 + rate / periods) ** (periods * years)
print("Amount:", amount)

# 4. Quadratic Equation Solver
def solve_quadratic():
    a = float(input("Enter coefficient a: "))
    b = float(input("Enter coefficient b: "))
    c = float(input("Enter coefficient c: "))
    discriminant = b**2 - 4*a*c
    if discriminant >= 0:
        root1 = (-b + math.sqrt(discriminant)) / (2*a)
        root2 = (-b - math.sqrt(discriminant)) / (2*a)
        print(f"The solutions are: {root1} and {root2}")
    else:
        root1 = ((-b)/2*a) + "i (" + (math.sqrt(abs(discriminant)) / 2*a) + ")"
        root2 = ((-b)/2*a) - "i (" + (math.sqrt(abs(discriminant)) / 2*a) + ")"
        print(f"The solutions are: {root1} and {root2}")

# 5. Logarithm Calculation
def calculate_logarithm():
    number = float(input("Enter the number: "))
    base = float(input("Enter the base: "))
    result = math.log(number, base)
    print(f"The logarithm of {number} to base {base} is {result}")

# 6. Exponential Growth Model
def exponential_growth():
    initial_amount = float(input("Enter the initial amount: "))
    growth_rate = float(input("Enter the growth rate: "))
    periods = int(input("Enter the number of periods: "))
    final_amount = initial_amount * math.exp(growth_rate * periods)
    print(f"The final amount is {final_amount}")

# 7. Trigonometric Functions
def trigonometric_functions():
    angle_degrees = float(input("Enter the angle in degrees: "))
    angle_radians = math.radians(angle_degrees)
    sine = math.sin(angle_radians)
    cosine = math.cos(angle_radians)
    tangent = math.tan(angle_radians)
    print(f"Sine: {sine}, Cosine: {cosine}, Tangent: {tangent}")

# 8. Distance between two points in 2D
def distance_2d():
    x1, y1 = map(float, input("Enter x1 and y1: ").split())
    x2, y2 = map(float, input("Enter x2 and y2: ").split())
    distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2)
    print(f"The distance between the points is {distance}")

# 9. Factorial Calculation
def calculate_factorial():
    number = int(input("Enter a non-negative integer: "))
    factorial = math.factorial(number)
    print(f"The factorial of {number} is {factorial}")
```

```
# 10. Angle Conversion
def angle_conversion():
    angle = float(input("Enter the angle: "))
    direction = input("Convert to (r)adians or (d)egrees? ").lower()
    if direction == 'r':
        converted_angle = math.radians(angle)
        print(f"{angle} degrees is {converted_angle} radians")
    elif direction == 'd':
        converted_angle = math.degrees(angle)
        print(f"{angle} radians is {converted_angle} degrees")

# 11. Surface Area and Volume of a Sphere
def sphere_calculations():
    radius = float(input("Enter the radius of the sphere: "))
    surface_area = 4 * math.pi * radius**2
    volume = (4/3) * math.pi * radius**3
    print(f"Surface area: {surface_area}, Volume: {volume}")

# 12. BMI Calculation
def calculate_bmi():
    weight = float(input("Enter your weight in kilograms: "))
    height = float(input("Enter your height in meters: "))
    bmi = weight / (height**2)
    print(f"Your BMI is {bmi}")

# 13. Celsius to Fahrenheit Conversion
def celsius_to_fahrenheit():
    celsius = float(input("Enter the temperature in Celsius: "))
    fahrenheit = (celsius * 9/5) + 32
    print(f"{celsius} degrees Celsius is {fahrenheit} degrees Fahrenheit")

# 14. Distance between two points in 3D
def distance_3d():
    x1, y1, z1 = map(float, input("Enter x1, y1, and z1: ").split())
    x2, y2, z2 = map(float, input("Enter x2, y2, and z2: ").split())
    distance = math.sqrt((x2 - x1)**2 + (y2 - y1)**2 + (z2 - z1)**2)
    print(f"The distance between the points is {distance}")

# 15. Simple Calculator
def simple_calculator():
    num1 = float(input("Enter the first number: "))
    num2 = float(input("Enter the second number: "))
    operation = input("Enter the operation (+, -, *, /): ")
    if operation == '+':
        result = num1 + num2
    elif operation == '-':
        result = num1 - num2
    elif operation == '*':
        result = num1 * num2
    elif operation == '/':
        result = num1 / num2
    else:
        print("Invalid operation")
        return
    print(f"The result is {result}")

# 16. Harmonic Mean
def harmonic_mean():
    numbers = list(map(float, input("Enter the numbers separated by spaces: ").split()))
    harmonic_mean_value = len(numbers) / sum(1/x for x in numbers)
    print(f"The harmonic mean is {harmonic_mean_value}")

# 17. Verify Trigonometric Identity
def verify_trig_identity():
    angle_degrees = float(input("Enter the angle in degrees: "))
    angle_radians = math.radians(angle_degrees)
    sin_squared = math.sin(angle_radians)**2
    cos_squared = math.cos(angle_radians)**2
    identity = sin_squared + cos_squared
    print(f"sin^2(x) + cos^2(x) = {identity}, which should be approximately 1")

# 18. Standard Deviation
def standard_deviation():
    numbers = list(map(float, input("Enter the numbers separated by spaces: ").split()))
    mean = sum(numbers) / len(numbers)
    variance = sum((x - mean)**2 for x in numbers) / len(numbers)
```

```
    std_deviation = math.sqrt(variance)
    print(f"The standard deviation is {std_deviation}")

# 19. GCD and LCM
def gcd_lcm():
    num1 = int(input("Enter the first number: "))
    num2 = int(input("Enter the second number: "))
    gcd = math.gcd(num1, num2)
    lcm = abs(num1 * num2) // gcd
    print(f"The GCD is {gcd} and the LCM is {lcm}")

# 20. Logarithmic Growth Model
def logarithmic_growth():
    initial_amount = float(input("Enter the initial amount: "))
    growth_rate = float(input("Enter the growth rate: "))
    periods = int(input("Enter the number of periods: "))
    final_amount = initial_amount * math.log(1 + growth_rate * periods)
    print(f"The final amount is {final_amount}")

# 21. Sine Wave Generation
def sine_wave():
    start_angle = float(input("Enter the start angle (in degrees): "))
    end_angle = float(input("Enter the end angle (in degrees): "))
    step_size = float(input("Enter the step size (in degrees): "))
    angles = [x for x in range(int(start_angle), int(end_angle) + 1, int(step_size))]
    sine_values = [math.sin(math.radians(angle)) for angle in angles]
    for angle, sine_value in zip(angles, sine_values):
        print(f"Sine({angle}) = {sine_value}")

# 22. Prime Number Check
def is_prime():
    number = int(input("Enter a number: "))
    if number <= 1:
        print(f"{number} is not a prime number")
        return
    for i in range(2, int(math.sqrt(number)) + 1):
        if number % i == 0:
            print(f"{number} is not a prime number")
            return
    print(f"{number} is a prime number")

# 23. Area of a Triangle using Heron's Formula
def area_of_triangle():
    a, b, c = map(float, input("Enter the lengths of the sides a, b, and c: ").split())
    s = (a + b + c) / 2
    area = math.sqrt(s * (s - a) * (s - b) * (s - c))
    print(f"The area of the triangle is {area}")

# 24. Future Value of an Investment
def future_value():
    principal = float(input("Enter the principal amount: "))
    annual_rate = float(input("Enter the annual interest rate (in decimal): "))
    years = int(input("Enter the number of years: "))
    future_value = principal * (1 + annual_rate)**years
    print(f"The future value of the investment is {future_value}")

# 25. Hyperbolic Functions
def hyperbolic_functions():
    number = float(input("Enter the number: "))
    sinh = math.sinh(number)
    cosh = math.cosh(number)
    tanh = math.tanh(number)
    print(f"Sinh: {sinh}, Cosh: {cosh}, Tanh: {tanh}")
```

```
['10', '12', '15', '6']
<map object at 0x7d65c1f3ceb0>
[10.0, 12.0, 15.0, 6.0]
The harmonic mean is 9.600000000000001
```

Assignment6: Simulate a Shoe Shop Using Python

Objective: Create a simulation of a shoe shop using Python. This assignment will cover various Python concepts, including object-oriented programming, the datetime module, and the random module. Requirements:

1. Shoe Class:

- Attributes: brand, size, price, stock, discount
- Methods: apply_discount(), restock(), sell(), return_shoe(), **str**()

2. ShoeShop Class:

- Attributes: inventory (a list of Shoe objects), sales, returns
- Methods: add_shoe(), remove_shoe(), buy_shoe(), return_shoe(), apply_store_discount(), generate_daily_report(), **str**()

3. Functional Requirements:

- Add shoes to inventory.
- Remove shoes from inventory.
- Buy shoes: Update inventory and sales, apply discount if available.
- Return shoes: Update inventory and returns.
- Apply discount to specific shoes.
- Apply a store-wide discount.
- Generate a daily sales report.

4. Utilize the datetime Module:

- Track the date and time of sales and returns.
- Generate reports for specific dates.

5. Utilize the random Module:

- Simulate random customer purchases and returns.
- Apply random discounts to shoes.

```
import random
from datetime import datetime

class Shoe:
    def __init__(self, brand, size, price, stock):
        self.brand = brand
        self.size = size
        self.price = price
        self.stock = stock
        self.discount = 0

    def apply_discount(self, discount):
        self.discount = discount

    def restock(self, amount):
        self.stock += amount

    def sell(self):
        if self.stock > 0:
            self.stock -= 1
            return self.price * (1 - self.discount)
        else:
            print("Out of stock!")
            return 0

    def return_shoe(self):
        self.stock += 1

    def __str__(self):
        return f"Brand: {self.brand}, Size: {self.size}, Price: ${self.price:.2f}, Stock: {self.stock}, Discount: {self.discount*100}%"

class ShoeShop:
    def __init__(self):
        self.inventory = []
        self.sales = []
        self.returns = []

    def add_shoe(self, shoe):
        self.inventory.append(shoe)

    def remove_shoe(self, shoe):
        if shoe in self.inventory:
            self.inventory.remove(shoe)

    def buy_shoe(self, brand, size):
        for shoe in self.inventory:
            if shoe.brand == brand and shoe.size == size:
                sale_price = shoe.sell()
                if sale_price > 0:
                    self.sales.append((shoe.brand, shoe.size, sale_price, datetime.now()))
                return
        print("Shoe not found in inventory.")

    def return_shoe(self, brand, size):
        for shoe in self.inventory:
            if shoe.brand == brand and shoe.size == size:
                shoe.return_shoe()
                self.returns.append((shoe.brand, shoe.size, shoe.price, datetime.now()))
                return
        print("Shoe not found in inventory.")

    def apply_store_discount(self, discount):
        for shoe in self.inventory:
            shoe.apply_discount(discount)

    def generate_daily_report(self):
        today = datetime.now().date()
        daily_sales = [sale for sale in self.sales if sale[3].date() == today]
        daily_returns = [return_ for return_ in self.returns if return_[3].date() == today]

        print("Daily Sales Report:")
        for sale in daily_sales:
            print(f"Brand: {sale[0]}, Size: {sale[1]}, Sale Price: ${sale[2]:.2f}, Date: {sale[3]}")

        print("\nDaily Returns Report:")
        for return_ in daily_returns:
            print(f"Brand: {return_[0]}, Size: {return_[1]}, Return Price: ${return_[2]:.2f}, Date: {return_[3]}")
```

```
    def __str__(self):
        inventory_str = "\n".join(str(shoe) for shoe in self.inventory)
        return f"Inventory:\n{inventory_str}\n"

# Example Usage
if __name__ == "__main__":
    shop = ShoeShop()

    # Adding shoes to the inventory
    shoe1 = Shoe("Nike", 42, 100, 10)
    shoe2 = Shoe("Adidas", 40, 90, 8)
    shoe3 = Shoe("Puma", 38, 80, 5)
    shop.add_shoe(shoe1)
    shop.add_shoe(shoe2)
    shop.add_shoe(shoe3)

    # Simulating random purchases and returns
    for _ in range(5):
        brand = random.choice(["Nike", "Adidas", "Puma"])
        size = random.choice([38, 40, 42])
        shop.buy_shoe(brand, size)

    for _ in range(2):
        brand = random.choice(["Nike", "Adidas", "Puma"])
        size = random.choice([38, 40, 42])
        shop.return_shoe(brand, size)

    # Applying a random discount
    for shoe in shop.inventory:
        discount = random.choice([0.1, 0.2, 0.3])
        shoe.apply_discount(discount)

    # Generating daily report
    shop.generate_daily_report()

    # Printing inventory
    print(shop)
```

Assignment 7: Fantasy Game Character Creation and Battle Simulation

Objective: Create a Python program that simulates the creation of fantasy game characters and their battles.

Requirements:

1. *Character Class:*
    - Attributes: name, health, attack, defense, level
    - Methods: attack_enemy(), defend(), take_damage(), level_up(), **str**()
2. Game Class:
    - Attributes: player, enemy
    - Methods: create_character(), simulate_battle(), log_battle()
3. Functional Requirements:
    - Create characters with random attributes (health, attack, defense) based on their level.
    - Simulate battles between a player character and an enemy character.
    - Log each battle with the date and time using the datetime module.
    - Implement leveling up mechanics based on battle outcomes.
    - Display character stats and battle results.
4. Utilize the datetime Module:
    - Log the date and time of each battle.
5. Utilize the random Module:
    - Generate random attributes (health, attack, defense) for characters.
    - Simulate random battle outcomes.

```
import random
from datetime import datetime

class Character:
    def __init__(self, name, level):
        self.name = name
        self.level = level
        self.health = random.randint(50, 100) * level
        self.attack = random.randint(10, 20) * level
        self.defense = random.randint(5, 15) * level

    def attack_enemy(self, enemy):
        damage = self.attack - enemy.defense
        damage = damage if damage > 0 else 0
        enemy.take_damage(damage)
        return damage

    def take_damage(self, damage):
        self.health -= damage
        if self.health < 0:
            self.health = 0

    def is_alive(self):
        return self.health > 0

    def level_up(self):
        self.level += 1
        self.health += random.randint(20, 40)
        self.attack += random.randint(5, 10)
        self.defense += random.randint(2, 5)

    def __str__(self):
        return (f"Name: {self.name}, Level: {self.level}, Health: {self.health}, "
                f"Attack: {self.attack}, Defense: {self.defense}")

class Game:
    def __init__(self):
        self.player = None
        self.enemy = None

    def create_character(self, name, level):
        return Character(name, level)

    def simulate_battle(self):
        if not self.player or not self.enemy:
            print("Both player and enemy must be created before starting a battle.")
            return

        print(f"Battle Start: {self.player.name} vs {self.enemy.name}")
        while self.player.is_alive() and self.enemy.is_alive():
            damage_to_enemy = self.player.attack_enemy(self.enemy)
            print(f"{self.player.name} attacks {self.enemy.name} for {damage_to_enemy} damage.")
            if self.enemy.is_alive():
                damage_to_player = self.enemy.attack_enemy(self.player)
                print(f"{self.enemy.name} attacks {self.player.name} for {damage_to_player} damage.")

        if self.player.is_alive():
            print(f"{self.player.name} wins!")
            self.player.level_up()
```