```
1)
# Initialize the total number of bugs collected
total bugs = 0
# Loop for five days
for day in range(1, 6):
  # Ask the user for the number of bugs collected that day
  bugs_collected = int(input(f"Enter the number of bugs collected on day {day}: "))
  # Add the number of bugs collected to the total
  total bugs += bugs collected
# Display the total number of bugs collected
print(f"Total number of bugs collected over five days: {total_bugs}")
2)
# Define the calories burned per minute
calories_per_minute = 4.2
# Define the time intervals in minutes
time intervals = [10, 15, 20, 25, 30]
# Initialize the index
index = 0
# Use a while loop to iterate through the time intervals
while index < len(time intervals):
  minutes = time_intervals[index]
  # Calculate calories burned
  calories_burned = calories_per_minute * minutes
  # Display the result
  print(f"Calories burned after {minutes} minutes: {calories_burned:.1f} calories")
  # Move to the next index
  index += 1
3)
# Get monthly budget from the user
budget = float(input("Enter your monthly budget: "))
# Initialize total expenses
total\_expenses = 0.0
# Collect expenses until the user types 'done'
while True:
  expense input = input("Enter an expense (or 'done' to finish): ")
```

```
if expense input.lower() == 'done':
     break
  total expenses += float(expense input)
# Calculate the remaining budget
difference = budget - total expenses
# Display the result
if difference > 0:
  print(f"You are under budget by: ${difference:.2f}")
else:
  print(f"You are over budget by: ${-difference:.2f}")
4)
# Get speed and hours from the user
speed = float(input("Enter the speed of the vehicle in mph: "))
hours = int(input("Enter the number of hours traveled: "))
# Print the header
print("Hour\tDistance Traveled")
# Calculate and display distance for each hour
for hour in range(1, hours + 1):
  print(f"{hour}\t{speed * hour}")
5)
# Get the number of years from the user
num years = int(input("Enter the number of years: "))
# Initialize total rainfall
total_rainfall = 0.0
# Loop through each year
for year in range(num years):
  print(f"\nYear {year + 1}:")
  # Loop through each month
  for month in range(1, 13):
     rainfall = float(input(f"Enter rainfall for month {month} (in inches): "))
     total rainfall += rainfall
# Calculate average rainfall
average rainfall = total rainfall / (num years * 12)
# Display results
print(f"\nTotal rainfall: {total rainfall:.2f} inches")
```

```
print(f"Average rainfall per month: {average rainfall:.2f} inches")
6)
# Print the table header
print("Celsius\tFahrenheit")
print("----")
# Loop through Celsius temperatures from 0 to 20
for celsius in range(21):
  fahrenheit = (9/5) * celsius + 32 # Convert Celsius to Fahrenheit
  print(f"{celsius}\t{fahrenheit:.2f}") # Print the Celsius and Fahrenheit values
7)
# Get the number of days from the user
num days = int(input("Enter the number of days: "))
# Initialize variables
total_pay = 0.0
daily salary = 0.01 # Starting salary in dollars (1 penny)
# Print the table header
print("Day\tSalary ($)")
print("----")
# Loop through each day to calculate and display the salary
for day in range(1, num days + 1):
  print(f"{day}\t{daily_salary:.2f}") # Display salary for the day
  total pay += daily salary # Accumulate total pay
  daily_salary *= 2 # Double the salary for the next day
# Display total pay at the end of the period
print("----")
print(f"Total pay over {num_days} days: ${total_pay:.2f}")
8)
# Get the number of days from the user
num_days = int(input("Enter the number of days: "))
# Initialize variables
total pay = 0.0
daily salary = 0.01 # Starting salary in dollars (1 penny)
# Print the table header
print("Day\tSalary ($)")
print("----")
# Loop through each day to calculate and display the salary
```

```
for day in range(1, num days + 1):
  print(f"{day}\t{daily_salary:.2f}") # Display salary for the day
  total pay += daily salary # Accumulate total pay
  daily salary *= 2 # Double the salary for the next day
# Display total pay at the end of the period
print("----")
print(f"Total pay over {num_days} days: ${total_pay:.2f}")
9)
# Define the rate of ocean level rise
rise_per_year = 1.6 # in millimeters
# Print the table header
print("Year\tOcean Level Rise (mm)")
print("----")
# Calculate and display the ocean level rise for each year
for year in range(1, 26): # 1 to 25 years
  total rise = rise per year * year # Total rise over the years
  print(f"{year}\t{total_rise:.1f}") # Print year and total rise
10)
# Initial tuition amount
initial tuition = 8000.0 # dollars
increase_rate = 0.03 # 3% increase
# Print the table header
print("Year\tProjected Tuition ($)")
print("----")
# Loop through the next 5 years to calculate and display the tuition
for year in range(1, 6):
  projected_tuition = initial_tuition * (1 + increase_rate) ** year
  print(f"{year}\t{projected_tuition:.2f}") # Print year and projected tuition
11)
# Get a nonnegative integer from the user
while True:
  try:
    n = int(input("Enter a nonnegative integer: "))
       print("Please enter a nonnegative integer.")
     else:
       break
  except ValueError:
```

```
print("Invalid input. Please enter an integer.")
# Initialize the factorial result
12)
# Get user input
starting population = int(input("Enter the starting number of organisms: "))
daily increase percentage = float(input("Enter the average daily increase (as a percentage): "))
num_days = int(input("Enter the number of days to multiply: "))
# Convert percentage to a decimal for calculation
daily increase rate = daily increase percentage / 100
# Print the table header
print("Day\tApproximate Population")
print("----")
# Initialize the population
current population = starting population
# Calculate and display the population for each day
for day in range(1, num days + 1):
  print(f"{day}\t{current_population:.6f}") # Display population with 6 decimal places
  current population *= (1 + daily increase rate) # Update population for the next day
13)
# Number of rows for the pattern
rows = 7
# Outer loop for each row
for i in range(rows, 0, -1):
  # Inner loop to print asterisks
  for j in range(i):
     print('*', end=") # Print asterisk without a newline
  print() # Move to the next line after printing asterisks
# Number of rows for the pattern
rows = 5
# Outer loop for each row
for i in range(rows):
  # Print the '#' for the first column
  print('#', end=")
  # Inner loop to print spaces and '#' for the remaining columns
  for i in range(i):
```

print(' #', end=") # Print space and '#' with the end parameter to stay on the same line

# Move to the next line after each row print()