Function Basics

1. Invoice Generator

Question:
Create a function `generate_invoice(items, tax_rate=0.07)` where `items` is a list of tuples with item names and prices, and `tax_rate` is an optional parameter with a default value of 7%. The function should return a formatted invoice string showing item names, their prices, the subtotal, the tax amount, and the total amount.

Sample Input:

```
items = [("Laptop", 1000), ("Mouse", 50), ("Keyboard", 75)]
print(generate_invoice(items))
```

Sample Output:

```
Invoice:
- Laptop: $1000
- Mouse: $50
- Keyboard: $75
Subtotal: $1125
Tax (7%): $78.75
Total: $1203.75
```

2. Password Strength Checker

Question:
Create a function `check_password_strength(password)` that evaluates the strength of a password based on criteria such as length, presence of uppercase letters, lowercase letters, digits, and special characters. The function should return a description indicating the strength of the password.

Sample Input:

```
print(check_password_strength("P@ssw0rd123"))
```

Sample Output:

```
Strong: Contains uppercase, lowercase, digit, and special character.
```

3. Weather Data Aggregator

Question:
Develop a function `aggregate_weather_data(city, days=7)` that retrieves weather data for a given city over a specified number of days (default to 7). The function should calculate and return the average temperature, humidity, and precipitation.

Sample Input:

```
weather_data = {
   "San Francisco": [
      [58, 60, 62, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84],  # Day 1
      [60, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84],  # Day 2
      [55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73,
74, 75, 76, 77, 78],  # Day 3
      [52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70,
71, 72, 73, 74, 75],  # Day 4
      [61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79,
80, 81, 82, 83, 84],  # Day 5
   ],
   "New York": [
      [75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57,
56, 55, 54, 53, 52],  # Day 1
      [77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59,
58, 57, 56, 55, 54],  # Day 2
      [80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62,
61, 60, 59, 58, 57],  # Day 3
      [82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64,
63, 62, 61, 60, 59],  # Day 4
      [79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67, 66, 65, 64, 63, 62, 61,
60, 59, 58, 57, 56],  # Day 5
   ],
   "Chicago": [
      [65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49, 48, 47,
46, 45, 44, 43, 42],  # Day 1
      [67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50, 49,
48, 47, 46, 45, 44],  # Day 2
      [68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51, 50,
49, 48, 47, 46, 45],  # Day 3
      [69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52, 51,
50, 49, 48, 47, 46],  # Day 4
      [70, 69, 68, 67, 66, 65, 64, 63, 62, 61, 60, 59, 58, 57, 56, 55, 54, 53, 52,
51, 50, 49, 48, 47],  # Day 5
   ],
   "Miami": [
      [85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67,
66, 65, 64, 63, 62],  # Day 1
      [87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69,
68, 67, 66, 65, 64],  # Day 2
      [88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70,
```

```
69, 68, 67, 66, 65],  # Day 3
      [90, 89, 88, 87, 86, 85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72,
71, 70, 69, 68, 67],  # Day 4
      [85, 84, 83, 82, 81, 80, 79, 78, 77, 76, 75, 74, 73, 72, 71, 70, 69, 68, 67,
66, 65, 64, 63, 62],  # Day 5
   ],
   "Los Angeles": [
      [70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88,
89, 90, 91, 92, 93

],  # Day 1
      [71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89,
90, 91, 92, 93, 94],  # Day 2
      [72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90,
91, 92, 93, 94, 95],  # Day 3
      [73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91,
92, 93, 94, 95, 96],  # Day 4
      [74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92,
93, 94, 95, 96, 97],  # Day 5
   ]
}

print(aggregate_weather_data("San Francisco", 5))
```

Sample Output:

```
Average Temperature: 65.6°F
Average Humidity: 68%
Average Precipitation: 0.2 inches
```

4. Stock Price Analyzer

Question:
Implement a function `analyze_stock_prices(stock_prices, threshold)` where `stock_prices` is a list of daily closing prices for a stock and `threshold` is a percentage. The function should return a list of dates where the price change exceeds the given threshold percentage.

Sample Input:

```
stock_prices = [100, 102, 98, 105, 107, 103]
print(analyze_stock_prices(stock_prices, threshold=5))
```

Sample Output:

```
Significant price changes:
- From day 1 to day 3: -4%
- From day 3 to day 4: +7%
```

5. Employee Salary Calculator

Question:
Write a function `calculate_salary(base_salary, hours_worked, overtime_rate=1.5)` that calculates an employee's total salary based on their base salary, hours worked, and an optional overtime rate. The function should assume that overtime is paid for hours worked beyond a standard 40-hour workweek.

Sample Input:

```
print(calculate_salary(50000, 45, overtime_rate=2.0))
```

Sample Output:

```
Total Salary: $50500
```

6. Text Summarizer

Question:
Create a function `summarize_text(text, max_sentences=3)` that takes a long piece of text and returns a summary consisting of up to `max_sentences` sentences.

Sample Input:

```
text = "Python is a high-level programming language. It is widely used for web
development, data analysis, and scientific computing. Python is known for its
readability and simplicity. It supports multiple programming paradigms and has a
vast ecosystem of libraries and frameworks."
print(summarize_text(text))
```

Sample Output:

```
Python is a high-level programming language. It is widely used for web
development, data analysis, and scientific computing. Python is known for its
readability and simplicity.
```

7. Movie Recommendation System

Question:
Develop a function `recommend_movies(user_preferences, movie_database)` where `user_preferences` is a dictionary of user preferences (e.g., genres, actors) and `movie_database` is a list of movies with their attributes. The function should recommend movies based on the user's preferences.

Sample Input:

```
user_preferences = {"genres": ["Action", "Adventure"], "actors": ["Tom Cruise"]}
movie_database = [
    {"title": "Mission: Impossible", "genres": ["Action", "Adventure"], "actors":
["Tom Cruise"]},
    {"title": "The Matrix", "genres": ["Sci-Fi", "Action"], "actors": ["Keanu
Reeves"]},
    {"title": "Top Gun", "genres": ["Action", "Drama"], "actors": ["Tom Cruise"]}
]
print(recommend_movies(user_preferences, movie_database))
```

Sample Output:

```
Recommended Movies:
- Mission: Impossible
- Top Gun
```

8. Expense Tracker

Question:
Implement a function `track_expenses(expenses, category=None)` where `expenses` is a list of tuples with expense categories and amounts. If a category is provided, the function should return the total expenses for that category; otherwise, return the total expenses for all categories.

Sample Input:

```
expenses = [("Food", 50), ("Transport", 20), ("Food", 30), ("Entertainment", 70)]
print(track_expenses(expenses, category="Food"))
```

Sample Output:

```
Total expenses for Food: $80
```

9. Language Translator

Question:
Create a function `translate_text(text, target_language="en")` that translates a given text to the specified target language (default is English).

Sample Input:

```
text = "Hola, ¿cómo estás?"
print(translate_text(text, target_language="en"))
```

Sample Output:

```
Hello, how are you?
```

10. Fitness Progress Tracker

Question:
Write a function `track_fitness_progress(activities)` where `activities` is a list of dictionaries with details about various fitness activities (e.g., running, cycling) including duration and calories burned. The function should return the total duration and calories burned, as well as a breakdown by activity type.

Sample Input:

```
activities = [
    {"type": "Running", "duration": 30, "calories_burned": 300},
    {"type": "Cycling", "duration": 45, "calories_burned": 400},
    {"type": "Swimming", "duration": 60, "calories_burned": 500}
]
print(track_fitness_progress(activities))
```

Sample Output:

```
Total Duration: 135 minutes
Total Calories Burned: 1200
Breakdown:
- Running: 30 minutes, 300 calories
- Cycling: 45 minutes, 400 calories
- Swimming: 60 minutes, 500 calories
```