CoGrammar

GitHub Workshop 2: Iteration and Functions

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(FBV: Mutual Respect.)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Open Classes. You can submit these questions here: Open Class Questions

Data Science Lecture Housekeeping cont.

- For all **non-academic questions**, please submit a query: www.hyperiondev.com/support

- Report a **safeguarding** incident: www.hyperiondev.com/safeguardreporting

- We would love your **feedback** on lectures: Feedback on Lectures

What does a for loop typically use in Python?

A. A counter that decreases with each iteration.
B. A condition that must always be true.
C. An iterable object to loop through.
D. A function call at each iteration.

When is a while loop most suitable in Python?

A. When the number of iterations is known beforehand.
B. When iterating over a list of fixed size.
C. When you need to loop until a certain condition is met.
D. When you need to iterate over a dictionary.

What are nested loops primarily used for?

A. To improve the speed of an algorithm.
B. To execute a loop within another loop.
C. To replace complex conditional statements.
D. To create a loop that never ends.

CoGrammar

Recap of Iteration

For Loops

- Used for iterating over items of a collection (like lists or strings) in the order that they appear.

While Loops

- Execute as long as a specified condition is true, useful for repeated actions until a condition changes.

Nested Loops

- A loop inside another loop, enabling the handling of multi-layered data structures.

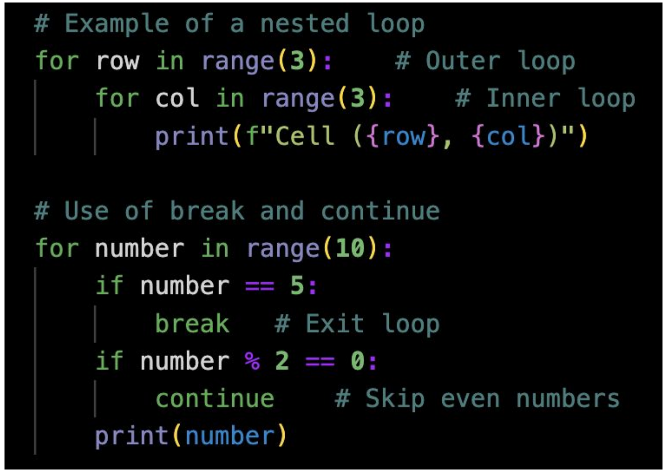
CoGrammar

Recap of Iteration

Controlling Loop Execution

- **‘break’:** Immediately exits the loop, typically used to exit early when a condition is met.

- **‘continue’:** Skips the rest of the current loop iteration and moves to the next one, often used to skip over certain items.



```
# Example of a nested loop
for row in range(3):    # Outer loop
    for col in range(3):    # Inner loop
        print(f"Cell ({row}, {col})")

# Use of break and continue
for number in range(10):
    if number == 5:
        break    # Exit loop
    if number % 2 == 0:
        continue    # Skip even numbers
    print(number)
```

CoGrammar

Recap of Functions

Defining functions

- In Python, user-defined functions are instantiated using the keyword ‘def’

Parameters

- These are variables that are defined in the function definition. They are assigned the values which were passed as arguments when the function was called, elsewhere in the code.

Return

- The values that a function returns when it completes.

Recap of Functions

Defining a Function

```
def add_one(x): # function called add_one
            y = x + 1
            return y
```

Calling a function

```
 result = add_one(5)
print(result)
```

Radiation Exposure Analysis

- **Background:** Dr. Eleanor is conducting comprehensive radiation studies in various environments to analyze radiation levels.

- **Challenge:** Managing a large dataset with readings from multiple locations and times, requiring complex data handling and analysis.

- **Objective:** Develop a programme to:
  - Input and store radiation data from different environments.
  - Calculate key statistics (average, standard deviation) to analyse data variability.
  - Identify patterns and anomalies across various datasets.

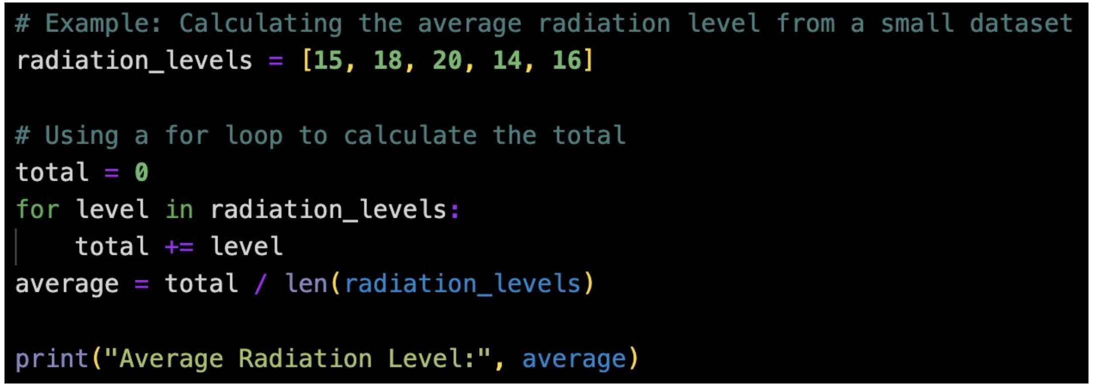
CoGrammar

Radiation Exposure Analysis

- **Programming Needs:**
  - Utilize for loops to iterate through datasets for each location.
  - Implement nested loops to process individual data points.
  - Incorporate while loops for ongoing data input until completion is signaled.

CoGrammar

Quick overview of Using Loops for Data Processing.



```python
# Example: Calculating the average radiation level from a small dataset
radiation_levels = [15, 18, 20, 14, 16]

# Using a for loop to calculate the total
total = 0
for level in radiation_levels:
    total += level
average = total / len(radiation_levels)

print("Average Radiation Level:", average)
```

We initialise a variable ‘total’ to 0, loop over the ‘radiation_levels’ to sum them up then divide it by the number of recorded levels to get the average.

CoGrammar

Demo: Data Organisation & Average Calculation

- Our data is stored in two Python Lists, one for the locations and another which stores the number of levels for each location

```python
# Define the lists to store locations and their radiation levels
locations = ["Urban", "Forest"]
levels = [[18, 20, 22], [12, 14, 15]]
```

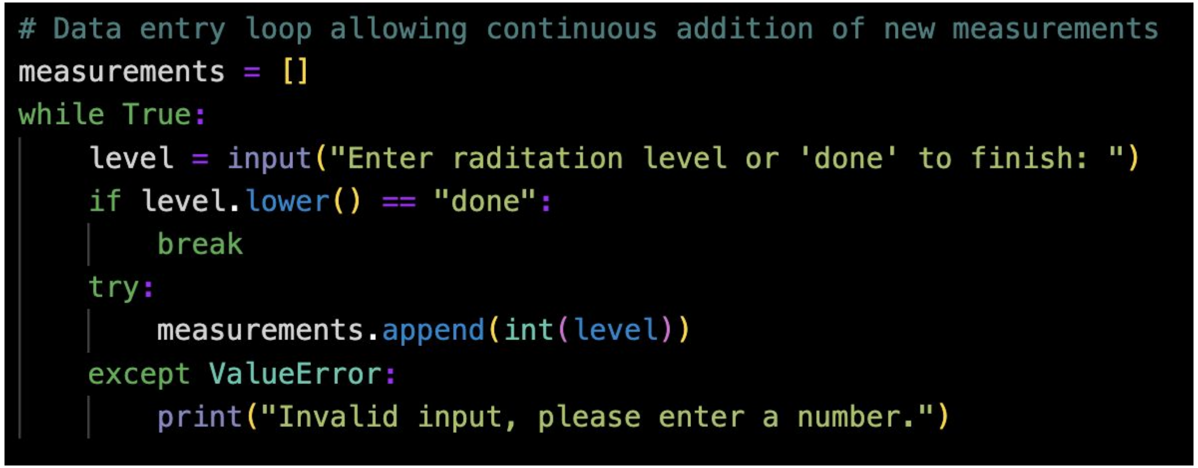
- To calculate the average of the levels we divide the sum of all the levels by the number of levels

```python
# Process each location's data and calculate the average radiation level
# enumerate function is used to get a counter when iterating through a list
for i, location in enumerate(locations):
    average = sum(levels[i]) / len(levels[i])
    print(f"{location} Average Radiation Level: {average}")
```

CoGrammar

Demo: Continuous Data Input

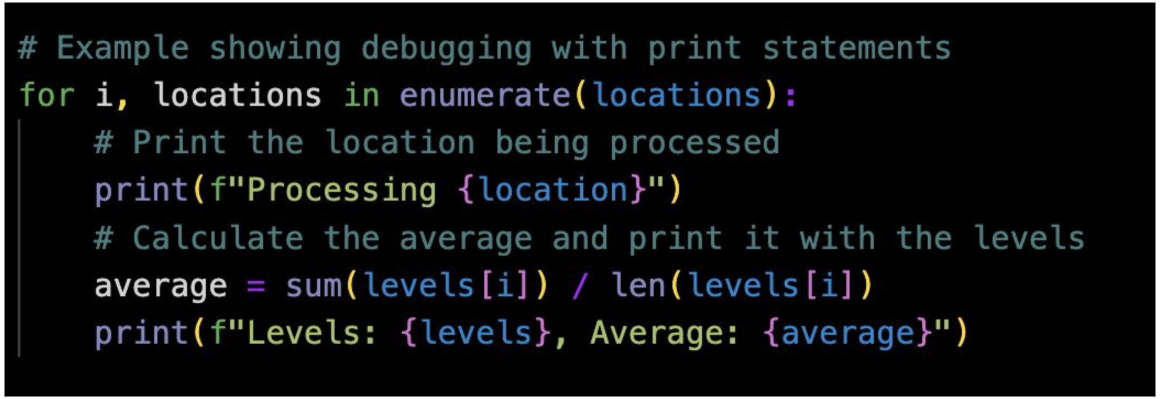
- In contrast to using a for loop that loops through elements in the dictionary, we could use a while loop that collects user measurements until they have given all the measurements and input “done”:



```python
# Data entry loop allowing continuous addition of new measurements
measurements = []
while True:
    level = input("Enter raditation level or 'done' to finish: ")
    if level.lower() == "done":
        break
    try:
        measurements.append(int(level))
    except ValueError:
        print("Invalid input, please enter a number.")
```

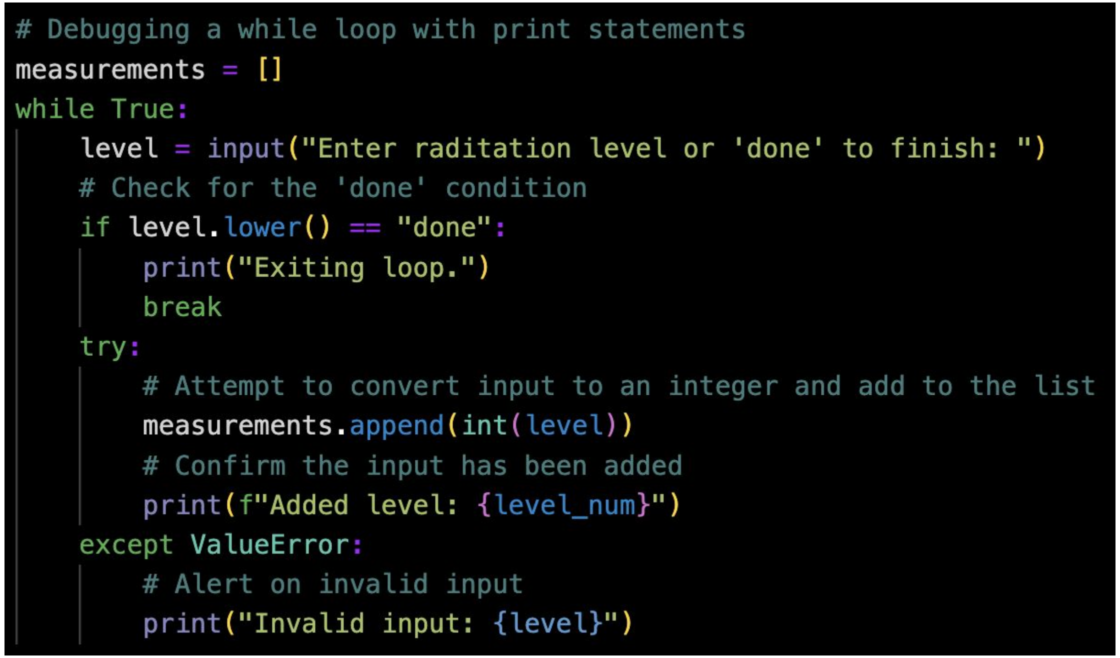
Demo: Debugging

- **Purpose of debugging:** Identify and correct errors in the code to ensure it runs as intended.
- Use print statements to monitor the values of variables during execution:



```python
# Example showing debugging with print statements
for i, locations in enumerate(locations):
    # Print the location being processed
    print(f"Processing {location}")
    # Calculate the average and print it with the levels
    average = sum(levels[i]) / len(levels[i])
    print(f"Levels: {levels}, Average: {average}")
```

- Check the control flow within loops to validate correct operation:



```python
# Debugging a while loop with print statements
measurements = []
while True:
    level = input("Enter raditation level or 'done' to finish: ")
    # Check for the 'done' condition
    if level.lower() == "done":
        print("Exiting loop.")
        break
    try:
        # Attempt to convert input to an integer and add to the list
        measurements.append(int(level))
        # Confirm the input has been added
        print(f"Added level: {level_num}")
    except ValueError:
        # Alert on invalid input
        print("Invalid input: {level}")
```

Dr. Eleanor’s Experiment

Your challenge is to calculate the average and standard deviation of radiation levels for multiple locations using Python loops.

Here is the data Dr. Eleanor wants you to use for developing the program:

Location	Radiation
City Center	[22, 19, 20, 31, 28]
Industrial Zone	[35, 32, 30, 37, 40]
Residential District	[15, 12, 18, 20, 14]
Rural Outskirts	[9, 13, 16, 14, 7]
Downtown	[25, 18, 22, 21, 26]

Step-by-Step Tasks:

1. **Data Organisation:** Organise data into a suitable structure.
2. **Average Calculation:** Decide on how you would calculate the average of multiple data observations.
3. **Standard Deviation Calculation:** Extend your program to calculate the standard deviation, giving insights into data variability.

Advanced Challenge:

- Once you’ve completed these steps add functionality for the user to input new radiation levels and integrate these into your calculations.

Tips:

- Use debugging to test each piece of your programme’s functionality as you develop it.

Summary

Fundamentals of Iteration

- ★ While loops are ideal for situations where the number of iterations is not predetermined, such as continuous user input.
- ★ Loops within loops useful to handle multi-layered data, crucial for complex data analysis tasks.
- ★ Explored how to use break to exit loops and continue to skip to the next iteration, providing greater control over loop execution.

Practical Problem Solving

- ★ Applied these concepts to solve a real-world problem: efficiently analyzing radiation levels across multiple locations, incorporating data organization, user interaction, and statistical calculations.

Summary

User-defined Functions

★ User-defined functions assist in encapsulating and reusing functionality in your program/code.

Higher-order Functions

★ Where functions take other functions as parameters or return functions as results.

In data processing, such as analysing radiation levels, what is a common use of nested for loops?

A. To execute different operations in parallel.
B. To process multi-dimensional data structures like lists of lists.
C. To stop execution after the first iteration.
D. To replace all conditional statements.

What is the purpose of the break statement in a loop in Python?

A. To execute different operations in parallel.
B. To process multi-dimensional data structures like lists of lists.
C. To stop execution after the first iteration.
D. To replace all conditional statements.

Questions and Answers

Questions around the Case Study