Introduction: Welcome to Europe! (10 mins)

Start by exploring a digital map of Europe together. Discuss its general location, size compared to other continents, and major surrounding bodies of water. Ask the student what they already know or find interesting about Europe. Point out distinct regions (e.g., Scandinavia, Mediterranean, Western/Eastern Europe) and maybe one or two major physical features like the Alps or the Danube River. Set the stage: 'Today, we're going on a coding adventure across Europe!'

Activity 1: Geographical Data Hunt (15 mins)

Using a reliable online atlas or map resource, the student's task is to research and list 10-15 European countries and their capitals. Encourage them to pick a mix from different regions. They should write these down neatly in their notebook or a digital document, ensuring correct spelling.

Activity 2: Introduction to Python Dictionaries (20 mins)

Explain that Python can help us store related pieces of information, like a country and its capital. Introduce Python dictionaries as containers for key-value pairs.
Analogy: Think of a real dictionary where the 'word' (key) leads you to its 'definition' (value). Here, the 'country' (key) will lead us to its 'capital' (value).
Guide the student to open their Python editor and create a new dictionary variable. Show the syntax:

```
european_capitals = {
  'Spain': 'Madrid',
  'France': 'Paris',
  'Germany': 'Berlin'
}
```

Have the student populate this dictionary with the 10-15 country-capital pairs they researched. Ensure they understand the syntax (quotes for strings, commas between pairs, curly braces). Practice accessing a value: `print(european_capitals['France'])`. Save the file (e.g., `europe_quiz.py`).

Activity 3: Coding the Euro-Quiz! (30 mins)

Now for the fun part - building the quiz! Guide the student step-by-step:

1. **Import Randomness:** At the top of the file, add `import random`. Explain this module lets Python make random choices.
2. **Get Countries:** We need a list of just the countries to pick from. Add: `countries = list(european_capitals.keys())`
3. **Pick a Random Country:** Add: `random_country = random.choice(countries)`
4. **Find the Correct Answer:** Get the capital for the chosen country: `correct_capital = european_capitals[random_country]`
5. **Ask the User:** Use the `input()` function to pose the question: `user_answer = input(f'What is the capital of {random_country}? ')`
6. **Check the Answer:** Use an `if/else` statement to compare the user's input with the correct answer. Introduce `.lower()` to make the comparison case-insensitive:

```
if user_answer.lower() == correct_capital.lower():
  print('Correct! Well done!')
else:
  print(f'Nice try! The correct capital is {correct_capital}.')
```

Encourage the student to type the code themselves, explaining each line as they go.

Activity 4: Test, Debug, Enhance! (15 mins)

Run the script multiple times. Did it work? If not, help the student debug common errors (typos, indentation, missing quotes/commas). Celebrate success!
Enhancement ideas (optional):

- Add more countries to the dictionary.
- Can they figure out how to put the quiz code inside a `while True:` loop to ask multiple questions? (Introduce `break` if needed).
- Could they add a score counter?

Wrap-up & Reflection (10 mins)

Review the countries and capitals covered. Discuss how Python helped organize the information and create an interactive learning tool. Ask reflection questions: What was the most challenging part? What was the most fun part? How else could you use programming to explore geography or other subjects?