Table of Contents

python import json

# 1 Assuming the function is available as parse\_code

sample\_code = ““” import os

class DataProcessor: def **init**(self, data): self.data = data

def process(self):  
 # A simple processing step  
 return len(self.data)

async def fetch\_data(url): print(f”Fetching from {url}“) return”some\_data” ““”

parsed\_objects = parse\_code(sample\_code) print(json.dumps(parsed\_objects, indent=2))

# 2 Example with a syntax error

invalid\_code = “def my\_func(:” result = parse\_code(invalid\_code) print(f”Result for invalid code: {result}“) ```

**Output:**

```json { “DataProcessor”: { “type”: “Class”, “name”: “DataProcessor”, “start\_line”: 3, “end\_line”: 9 }, “**init**”: { “type”: “Function”, “name”: “**init**”, “start\_line”: 4, “end\_line”: 5 }, “process”: { “type”: “Function”, “name”: “process”, “start\_line”: 7, “end\_line”: 9 }, “fetch\_data”: { “type”: “Function”, “name”: “fetch\_data”, “start\_line”: 11, “end\_line”: 13 } } Result for invalid code: {}

python # Define a variable named ‘name’ name = “Alice”

# 3 The following line will print the string “Alice” to the console.

print(name)

# 4 Example with a different data type

name = 42 # This will print the number “42” to the console. print(name)

python # Assuming the function is in a file named utils.py # from utils import fibonacci

def fibonacci(n): ““” Calculates the n-th Fibonacci number using an iterative approach. ““” if n <= 0: return 0 elif n == 1: return 1 else: a, b = 0, 1 for \_ in range(2, n + 1): a, b = b, a + b return b

# 5 Calculate the 10th Fibonacci number

fib\_10 = fibonacci(10) print(f”The 10th Fibonacci number is: {fib\_10}“)

# 6 Test edge cases

fib\_0 = fibonacci(0) fib\_1 = fibonacci(1) print(f”The 0th Fibonacci number is: {fib\_0}“) print(f”The 1st Fibonacci number is: {fib\_1}“)