Python Advance

Lesson 2

Revisit

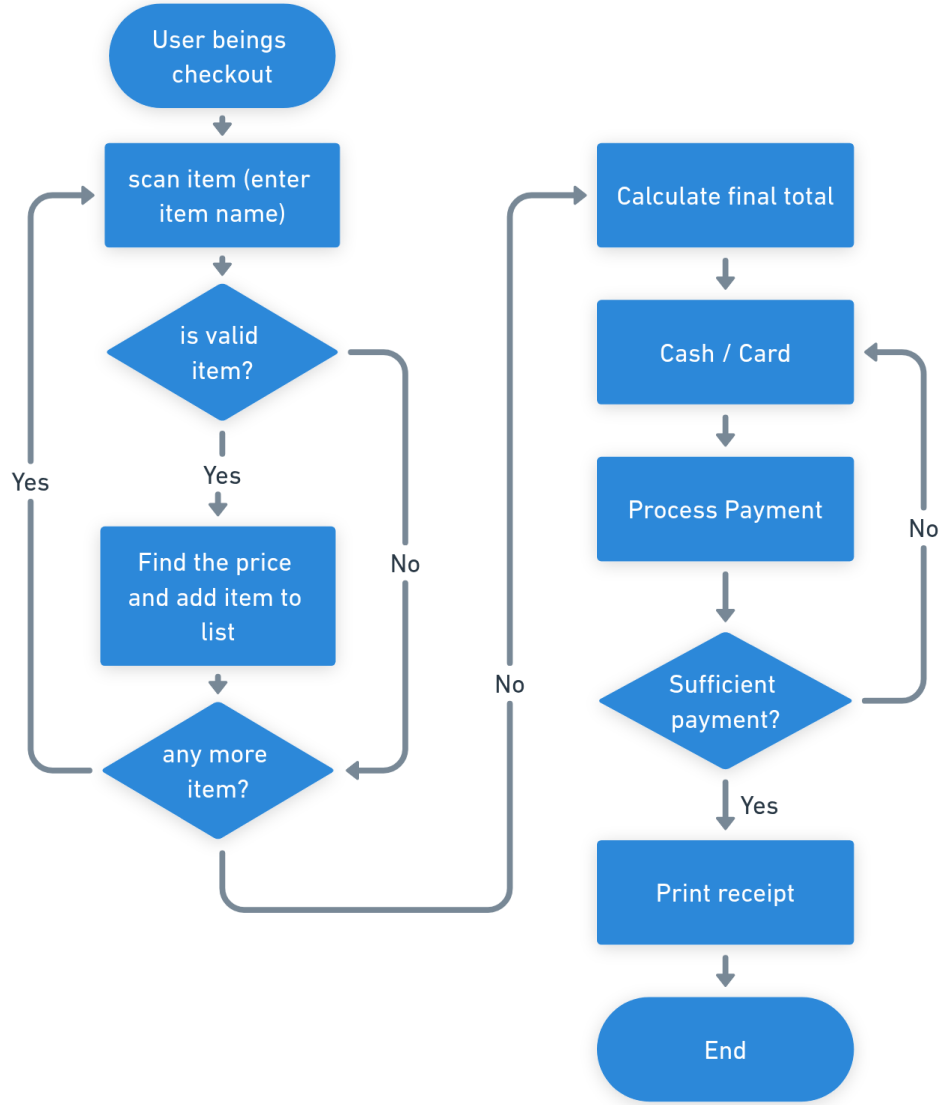
- Function Revisit
- Software Design – a simple principle
- Flowchart
- Modular Design

Quiz – Data Types

https://pynative.com/python-variables-and-data-types-quiz/

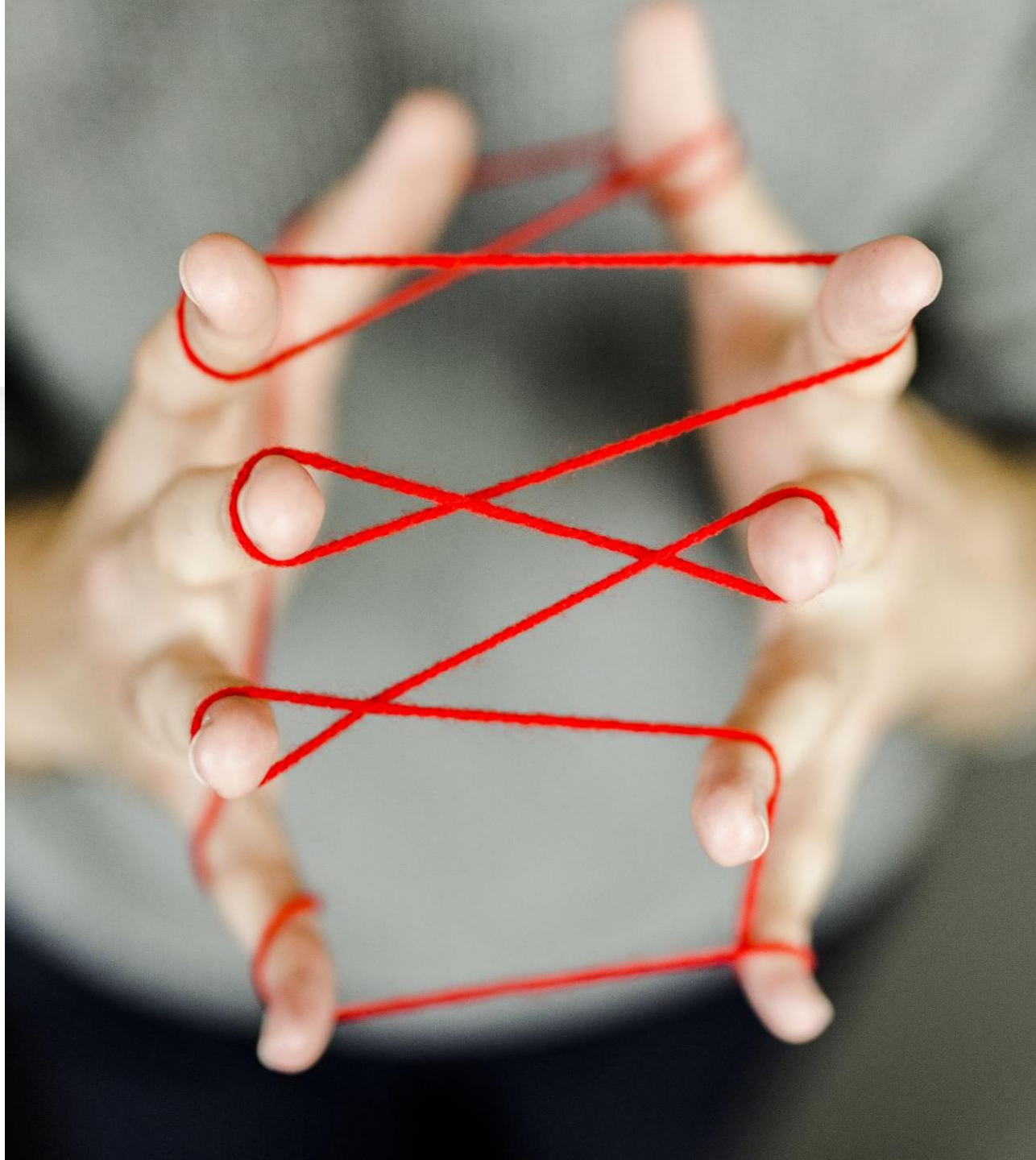
Exercise Review

- Flowchart



Exercise Review

- Modular Design - Detail Explanations



Error Handling

- When error occurred, system stop the execution
- How do we handle this?

```
1 print("===== Welcome to BMI Calculator =====")
2 weight = float(input("Your weight in kg : "))
3 height = float(input("Your eight in cm : "))
4 print(f"Your BMI indicator is {weight/height}")
5
```

Running: test.py

```
===== Welcome to BMI Calculator =====
Your weight in kg : 70
Your eight in cm : 0
Traceback (most recent call last):
  File "c:\users\quinc\mu_code\test.py", line 4, in <module>
    print(f"Your BMI indicator is {weight/height}")
ZeroDivisionError: float division by zero
>>>
```

Lab

- Create a simple calculator that won't crash when users enter invalid input.
- Asks the user for two numbers
- Asks for an operation (+, -, *, /)
- Performs the calculation
- Handles errors gracefully
- Error types to handle:
    1. ValueError - Invalid number input
    2. ZeroDivisionError - Division by zero
    3. User input validation - Invalid operation symbols
- Hints: which parts of the code above might generate errors?

JavaScript Object Notation (JSON)

- JSON is a way to store and exchange data.
- Think of it like a digital filing cabinet where information is organized in a specific format.
- JSON uses a format similar to Python dictionaries and lists
- Sample format:

```
{
  "name": "Alice",
  "age": 12, "hobbies": ["reading", "coding", "gaming"],
  "has_pet": true,
  "favorite_number": 7
}
```

Reading JSON Data – Single Record

- Read JSON content

```
import json

# JSON string (what we get from APIs)
json_string = '{"student": "John", "grade": 8, "subjects": ["Math", "Science"]}'

# Convert JSON to Python dictionary
data = json.loads(json_string)
print(data["student"])  # Output: John
print(data["subjects"][0])  # Output: Math
```

Reading JSON Data – Multiple Records

```
import json

school_json = '''
{
    "school_name": "KidsCode Academy",
    "students": [
        {"name": "John", "age": 13, "grade": 8},
        {"name": "Alice", "age": 14, "grade": 9},
        {"name": "Bob", "age": 12, "grade": 7}
    ],
    "teachers": [
        {"name": "Mr. Smith", "subject": "Math", "years_experience": 5},
        {"name": "Ms. Johnson", "subject": "Science", "years_experience": 8}
    ],
    "courses": {
        "programming": ["Python", "JavaScript", "HTML"],
        "mathematics": ["Algebra", "Geometry"],
        "science": ["Physics", "Chemistry", "Biology"]
    }
}
'''
```

Reading JSON Data – Multiple Records

```
school_data = json.loads(school_json)

# Iterate through students
print("STUDENTS:")
for student in school_data["students"]:
    print(f"- {student['name']}, Age: {student['age']}, Grade: {student['grade']}")

print("\nTEACHERS:")
for teacher in school_data["teachers"]:
    print(f"- {teacher['name']} teaches {teacher['subject']} ({teacher['years_experience']}
years exp.)")

print("\nCOURSES:")
for category, course_list in school_data["courses"].items():
    print(f"- {category.title()}: {', '.join(course_list)}")
```

Reading JSON Data – Output

```
STUDENTS:
- John, Age: 13, Grade: 8
- Alice, Age: 14, Grade: 9
- Bob, Age: 12, Grade: 7

TEACHERS:
- Mr. Smith teaches Math (5 years exp.)
- Ms. Johnson teaches Science (8 years exp.)

COURSES:
- Programming: Python, JavaScript, HTML
- Mathematics: Algebra, Geometry
- Science: Physics, Chemistry, Biology
```

Application Programming Interface (API)

Explanation

Exercise

1. Choose ONE API from the list above
2. Write a Python program that:
   - Makes a request to the API
   - Handles potential errors (network issues, invalid responses)
   - Displays the returned data in a user-friendly format
   - Allows users to make multiple requests

- Hints: refer to the API explanation page

Exercise – Sample Code Structure

```
import requests
import json

def get_api_data():
    try: # Make API request
        response = requests.get("YOUR API URL")
        if response.status_code == 200:
            data = response.json() # Process and display data
            if (data):
                return data
        else:
            print("API request failed")
    except Exception as e:
        print(f"Error: {e}")

# Main program loop to be enhanced:
#    print user-friendly format data;
#    allow user to make multiple requests
data = get_api_data()
print(data)
```

APIs for Exercise

API	Document	Sample API URL
Joke	https://sv443.net/jokeapi/v2/	https://v2.jokeapi.dev/joke/Any
Numbers	http://numbersapi.com/#42	http://numbersapi.com/random/year?json
Quote	https://docs.zenquotes.io/zenquotes-documentation/	https://zenquotes.io/api/random