CoGrammar

SE PORTFOLIO SESSION 9

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

Software Engineering Lecture Housekeeping

- The use of disrespectful language is prohibited if asking a question. This is a supportive, learning environment for all – please engage accordingly! **(FBV: Mutual Respect.)**
- No question is ‘silly’ – **ask away!**
- 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**

Software Engineering 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

Progression Criteria

✅ **Criterion 1: Initial Requirements**

- Complete 15 hours of Guided Learning Hours and the first four tasks within two weeks.

✅ **Criterion 2: Mid-Course Progress**

- Software Engineering: Finish 14 tasks by week 8.
- Data Science: Finish 13 tasks by week 8.

✅ **Criterion 3: Post-Course Progress**

- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

✅ **Criterion 4: Employability**

- Record a Final Job Outcome within 12 weeks of graduation, or by 23rd September 2024.

True or False: A function can be passed to another function as an argument?

A. True
B. False

What is a decorator in Python used for in functions?

A. A function that adds functionality to another function
B. A built-in Python module
C. A data type for function parameters
D. An alias for a function

Recap of Week 9: Functions

Default Values

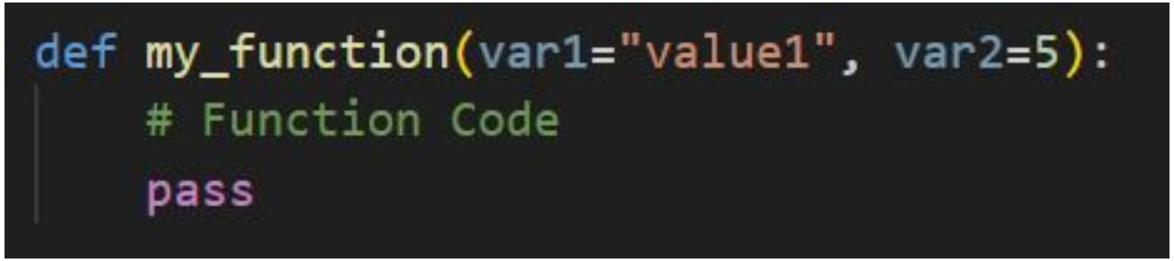
- We can give our parameter variables default values they can revert to if no argument is given.

Function Wrappers

- We can create function wrappers that allow us to execute given code before executing our function.

Recap of Week 9: Functions

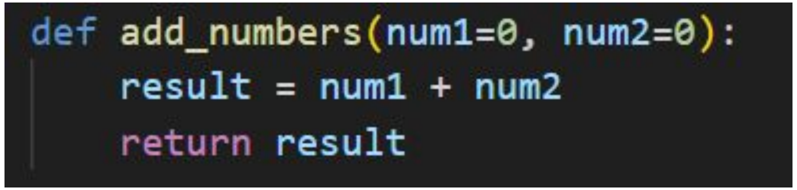
Adding Default Values



```
def my_function(var1="value1", var2=5):
    # Function Code
    pass
```

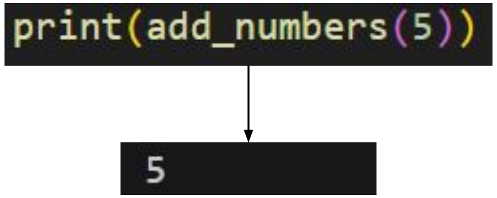
Recap of Week 9: Functions

Adding Default Values



```python
def add_numbers(num1=0, num2=0):
    result = num1 + num2
    return result
```

Calling Function

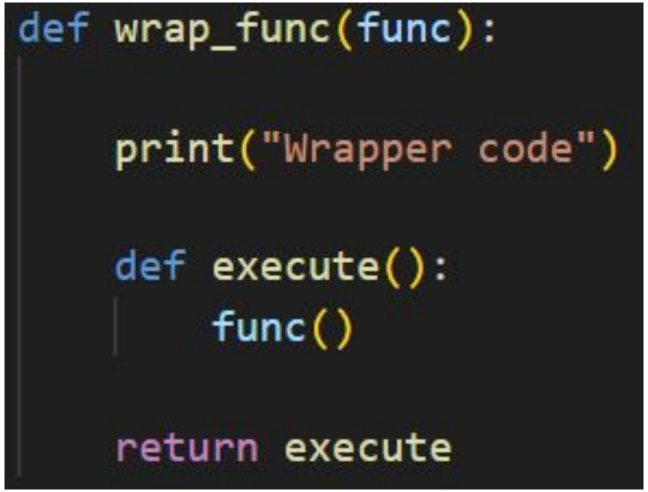


```python
print(add_numbers(5))
```

```
5
```

Recap of Week 9: Functions

Creating a Function Wrapper

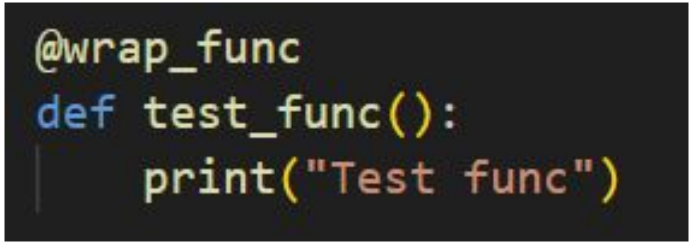


```
def wrap_func(func):

    print("Wrapper code")

    def execute():
        func()

    return execute
```

CoGrammar

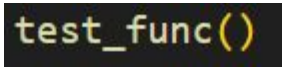
Recap of Week 9: Functions

Wrapping a Function



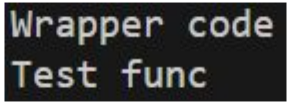
```
@wrap_func
def test_func():
    print("Test func")
```

Calling the Function



```
test_func()
```

Output



```
Wrapper code
Test func
```

Fitness Program

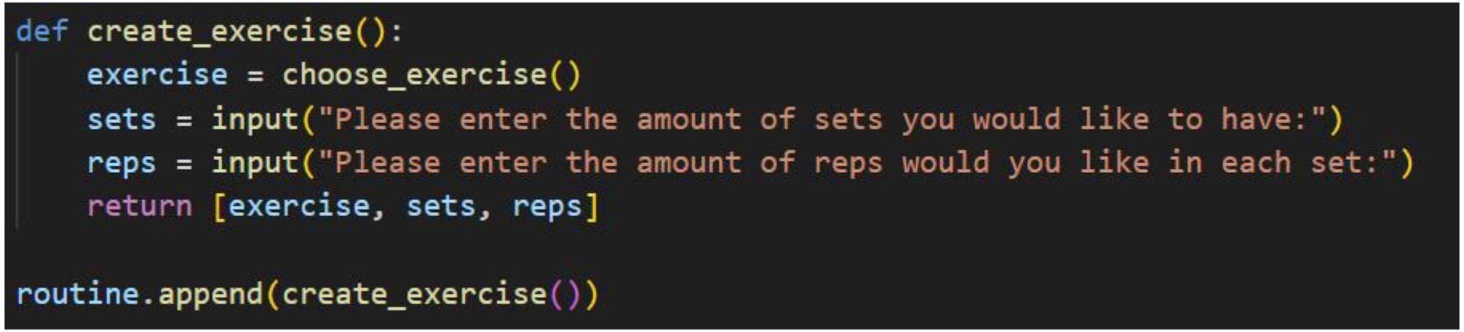
- **Background:** Tomasz is a fitness coach committed to assisting his clients in reaching their wellness objectives. His goal is to develop a fitness program that lets users design custom diets and exercise regimens using user-defined features

- **Challenge:** You are tasked with creating a fitness program that generates personalised workout routines and nutrition plans, tracks client progress such as meals and fitness goals, and provides them with instructions on how to follow the plan and do the required workouts.

- **Objective:**
  - Create user-defined functions for generating personalised workout routines and nutrition plans.
  - Track client progress, meals, and fitness goals.
  - Provide good workout instructions and client communication.

Creating Exercises

We can create a function that will prompt for the required input and build a exercise for us in the form of a list. We can then have a routine list to store all of our exercises in.

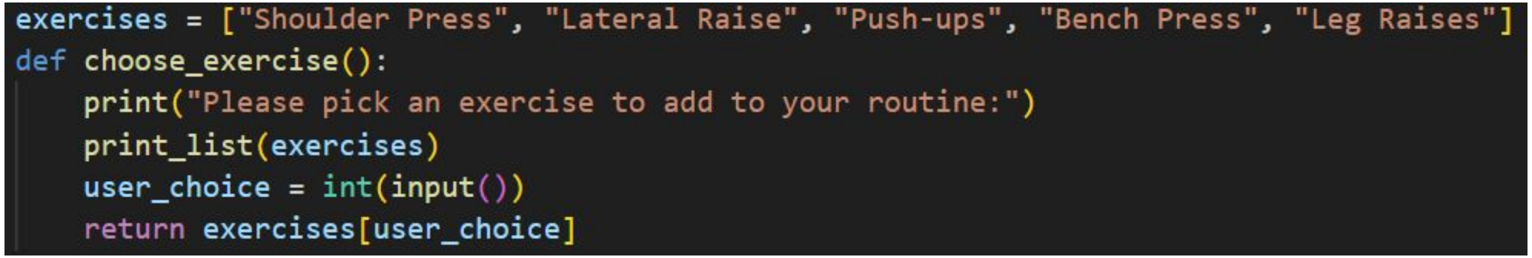


```python
def create_exercise():
    exercise = choose_exercise()
    sets = input("Please enter the amount of sets you would like to have:")
    reps = input("Please enter the amount of reps would you like in each set:")
    return [exercise, sets, reps]

routine.append(create_exercise())
```

CoGrammar

Creating Exercises

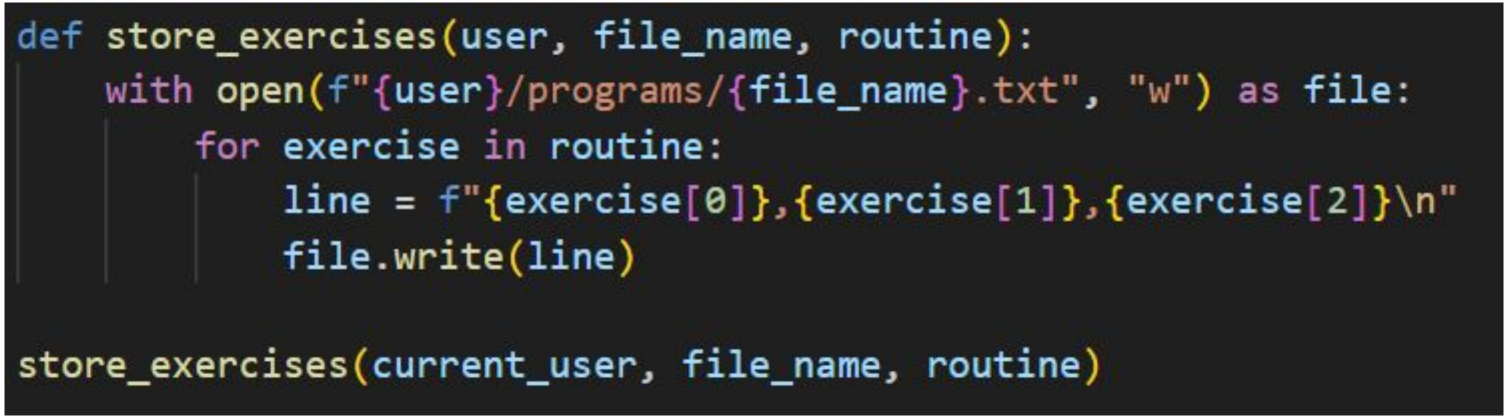
The choose_exercise() function lists all available exercises for the user to select one and returns the selected exercise.



```python
exercises = ["Shoulder Press", "Lateral Raise", "Push-ups", "Bench Press", "Leg Raises"]
def choose_exercise():
    print("Please pick an exercise to add to your routine:")
    print_list(exercises)
    user_choice = int(input())
    return exercises[user_choice]
```

Creating Exercises

Now that we have our routine stored in a variable, we can store the data in a text file for later use. This function takes a username, a filename and the routine as arguments. The user and file names are used to build a path for storing the file, and the routine contains all the data we need to output to the file.

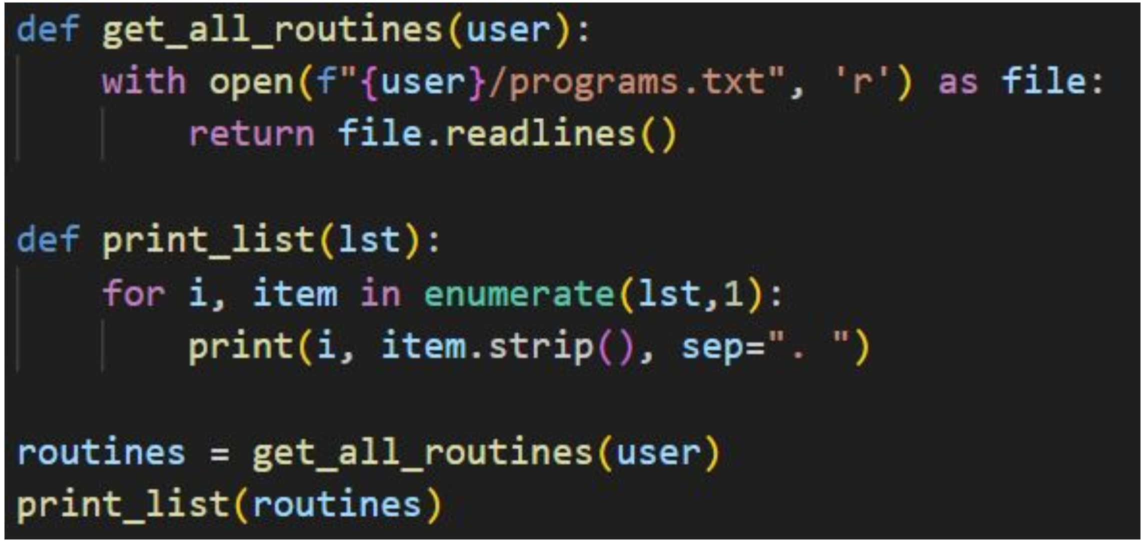


```python
def store_exercises(user, file_name, routine):
    with open(f"{user}/programs/{file_name}.txt", "w") as file:
        for exercise in routine:
            line = f"{exercise[0]},{exercise[1]},{exercise[2]}\n"
            file.write(line)

store_exercises(current_user, file_name, routine)
```

CoGrammar

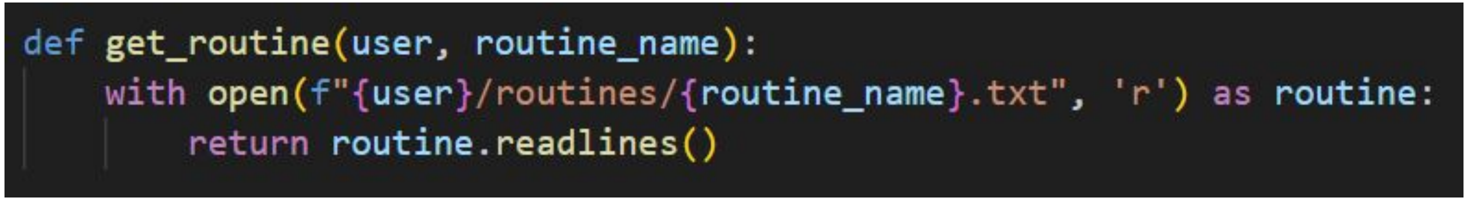
Retrieving Routines

To get all the routines a user created we can store all the routine names in a text file. We can then print out the names of all the routines to allow the user to choose one.



```python
def get_all_routines(user):
    with open(f"{user}/programs.txt", 'r') as file:
        return file.readlines()

def print_list(lst):
    for i, item in enumerate(lst,1):
        print(i, item.strip(), sep=". ")

routines = get_all_routines(user)
print_list(routines)
```

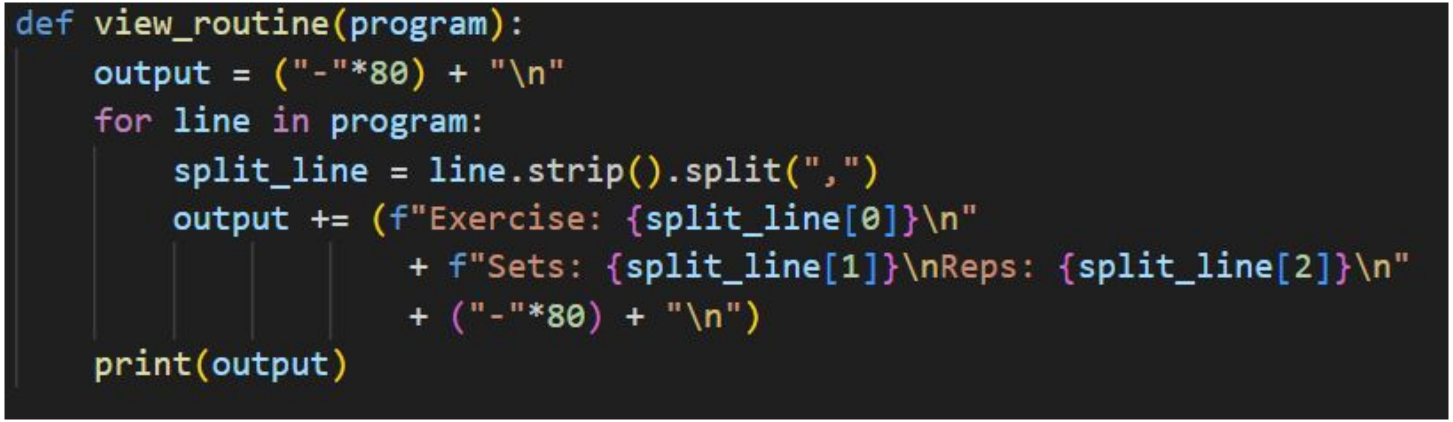
Retrieving Routines



```python
def get_routine(user, routine_name):
    with open(f"{user}/routines/{routine_name}.txt", 'r') as routine:
        return routine.readlines()
```

We can retrieve a routine using the user's username and the name of the routine. Using the username and routine name we can navigate to the correct path to get to the text file containing all our data.

Retrieving Routines



```python
def view_routine(program):
    output = ("-"*80) + "\n"
    for line in program:
        split_line = line.strip().split(",")
        output += (f"Exercise: {split_line[0]}\n"
                    + f"Sets: {split_line[1]}\nReps: {split_line[2]}\n"
                    + ("-"*80) + "\n")
    print(output)
```

We can then view the routine by printing out all the exercises in a neat and readable manner.

Fitness Program

You are tasked with creating a fitness program generates personalised workout routines and nutrition plans, track client progress such as meals, and fitness goals and provide them with instructions on how to follow the plan and do the required workouts.

Important features:

1. **Menu:** Provide the user with a menu listing all the available option such as creating a new routine, editing a routine, starting a routine, etc.
2. **Building Routines:** The user should be able to build exercise routines where they can add, remove, and edit their routines.
3. **Diet Plans:** Allow the user to build diet plans that they can track using the application.
4. **Workout Instructions:** Provide the user with instructions on how to follow the routine and diet plan as well as how to perform the required exercises.

Advanced Challenge:

- Allow users to filter their choices when selecting an exercise, e.g., if a user is looking for shoulder exercises the program should only show the available shoulder exercises.

Summary

Fitness Program

★ Create a fitness program that will build and track workout routines and diet plans.

Functions

★ Create all the required functionality using your own defined functions.

Modular & Modifiable

★ Make sure your functions a modular and modifiable for future changes.

User Experience

★ Remember to always keep the user experience in mind as users do not want to work on difficult programs.

In Python, what is a lambda function?

A. A function with no return statement
B. An anonymous function
C. A function with only one parameter
D. A function without any arguments

What does the *args syntax in a function definition represent?

A. A tuple of arguments
B. A list of arguments
C. An optional argument
D. Allowing any number of keyword arguments

Questions and Answers

Questions around the Case Study