



CoGrammar

GitHub Workshop

**SKILLS
FOR LIFE**

SKILLS BOOTCAMPS



Department
for Education

Software Engineering Lecture Housekeeping


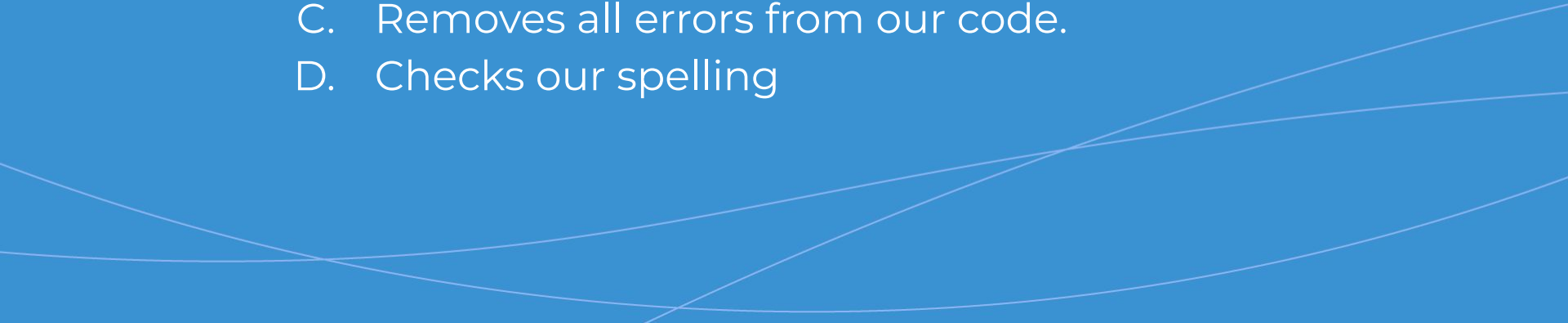
- The use of disrespectful language is prohibited if asking a question. This is a supportive, learning environment for all – please engage accordingly! **(FBV: Mutual Respect.)**
- No question is ‘silly’ – **ask away!**
- There are **Q&A sessions** midway and at the end of the session, should you wish 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](#)

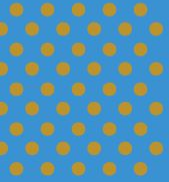
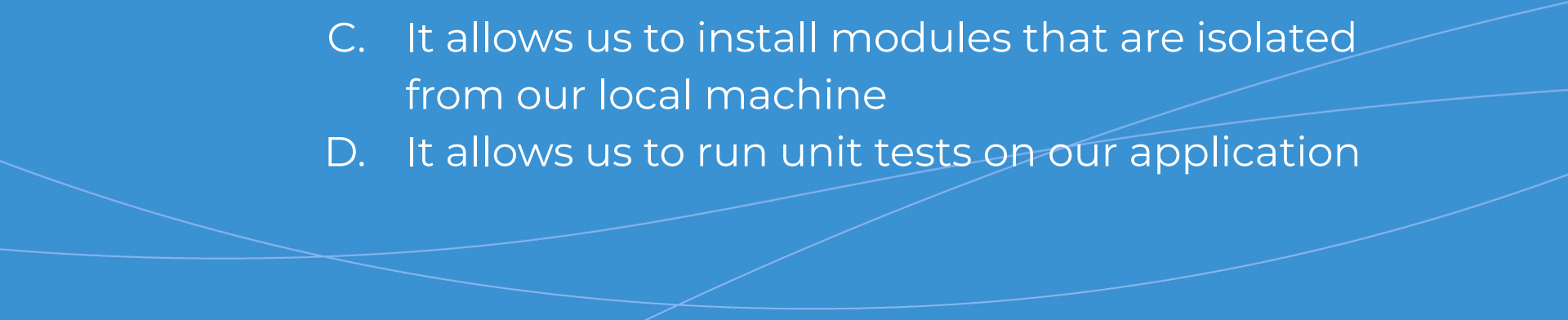


What does a Linter do?

- 
- A. Tests the functionality of our code
 - B. Warns us about coding standards violations
 - C. Removes all errors from our code.
 - D. Checks our spelling
- 



Why do we use virtual environments?

- 
- A. It allows us to connect to GitHub
 - B. We can mimic another operating system from our local machine
 - C. It allows us to install modules that are isolated from our local machine
 - D. It allows us to run unit tests on our application
- 

Recap of Week: Unit Tests

What Are They

- Automated tests that allow us to test the functionality of our code.

Why Use Them?

- Gives developer the confidence to make changes to their code.
- Allows us to validate the requirements for our application.

Recap of Week: Virtual Environment

What is it

- Built in Python package that allows us to isolate our application dependencies
- Allows each project to store it's dependencies without needing to link to locally installed modules.

Why Use Them?

- Each project can have the correct version for the required modules
- We can test new modules without affecting our local installs.

Recap of Week: Git and GitHub

What Are They?

- Tools for keeping a history of our project locally and remotely
- We can take snapshots of different points in our project history using commits.

Why Use Them?

- Makes managing the projects a lot easier when everything is stored on a single repository
- We can go back in time to past commits when we need to
- We can use branches to test new features without affecting our working application

Recap of Week: Linting

What is it?

- Lets the developer know if their work aligns with coding standards

Why Use Them?

- Aligning with standards makes the code easier to work with when in a team
- When code is being reviewed, reviewers don't need to worry about coding standards and just focus on functionality.

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:**
 - Set up your development environment
 - Set up a GitHub Repository for the project
 - Plan your solution
 - Based on your planning, create the file structure and implement the code
 - Create unit tests for as many parts of the application as possible
 - Continuously test, commit and push your code.

Creating Exercises

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

```
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 you would like in each set:")  
    return [exercise, sets, reps]  
  
routine.append(create_exercise())
```

Creating Exercises

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

```
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.

```
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)
```

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.

```
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

```
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

```
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.

Separation of Concern

- ★ Make sure that you are breaking the code up so that each concern is being handled individually

Modular & Modifiable



- ★ Make sure that the code is easy to manage and update, remember that tests make this possible long with modular code.

User Experience

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



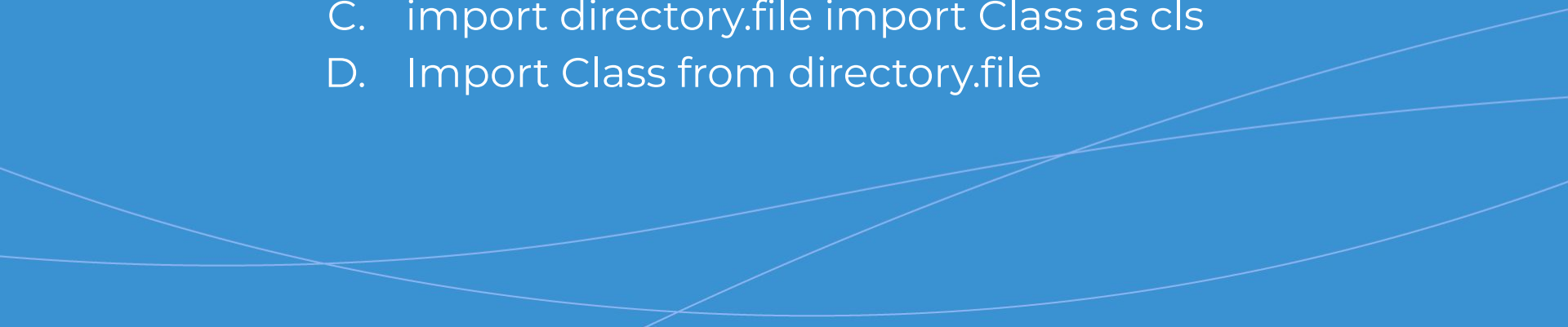
How often should you push your work to GitHub?

- 
- A. Once a day
 - B. After each line of code
 - C. After each commit
 - D. Every minute
- 



Which of the following is not the correct order for importing modules?



- A. `from directory.file import Class`
 - B. `Import directory.file`
 - C. `import directory.file import Class as cls`
 - D. `Import Class from directory.file`
- 



Questions and Answers

Questions around the Case Study

