



DS PORTFOLIO SESSION 7

**SKILLS
FOR LIFE**

SKILLS BOOTCAMPS



Department
for Education

Data Science Lecture Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly.
(FBV: Mutual Respect.)
- No question is daft or silly - **ask them!**
- 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](#)

Data Science 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.

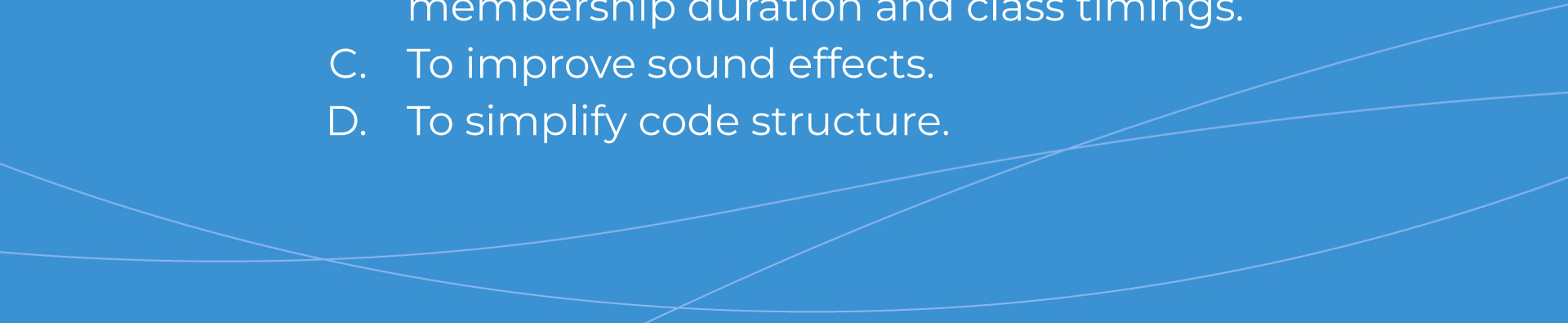
✓ **Criterion 4: Employability**

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



Why is parameterisation important in functions?



- A. To enhance graphics.
 - B. To cater to individual user preferences, such as membership duration and class timings.
 - C. To improve sound effects.
 - D. To simplify code structure.
- 

Recap of Week 7: Functions

Defining functions

- In Python, user-defined functions are instantiated using the keyword `'def'`

Parameters

- These are variables that are defined in the function definition. They are assigned the values that were passed as arguments when the function was called, elsewhere in the code.

Return

- The values that a function returns when it completes.

Recap of Week 7: Functions

Defining a Function

```
def add_one(x): # function called add_one
    y = x + 1
    return y
```

Calling a function

```
result = add_one(5)
print(result)
```

Gym Management System

- **Background:** You are part of the software development team at Starship Logistics; a logistics company specialising in shipping goods globally. You have decided to create and implement a comprehensive log management system for tracking and analysing the vast amount of shipping-related logs that the company deals with.
- **Challenge:** Develop the FitLife Gym Membership Management System. This system will handle member registrations, class scheduling, and membership renewals.
- **Objective:**
 - Utilise functions to manage memberships, class scheduling and renewals.

Demo: Calculating Membership

```
# Simple example to demonstrate functions in FitLife Gym Membership
Management System
def calculate_membership_price(package, duration):
    base_price = 50 # Base price for a monthly membership

    if package == 'quarterly':
        base_price *= 3
    elif package == 'yearly':
        base_price *= 12

    return base_price * duration

# Call the function
membership_cost = calculate_membership_price('monthly', 6)
print(f"The total cost for a 6-month monthly membership is ${membership_cost}.")
```

Demo: Higher Order Functions

```
# Extended example to demonstrate higher-order functions in FitLife Gym
Membership Management System

def apply_discount(package, original_price, discount_function):
    discounted_price = discount_function(original_price)
    return f"The discounted price for a {package} membership is
    ${discounted_price}."

def percentage_discount(original_price):
    return original_price * 0.9 # 10% discount

def fixed_amount_discount(original_price):
    return original_price - 5 # $5 fixed amount discount

# Call higher-order functions
monthly_discounted = apply_discount('monthly', 50, percentage_discount)
yearly_discounted = apply_discount('yearly', 600, fixed_amount_discount)

print(monthly_discounted)
print(yearly_discounted)
```

FitLife

Develop the FitLife Gym Membership Management System. This system will handle member registrations, class scheduling, and membership renewals.

Here is a list of some of the methods for your program.

<code>calculate_membership_price()</code>
<code>apply_discount()</code>
<code>percentage_discount()</code>
<code>fixed_amount_discount()</code>
Conversion Functions
<code>find()</code>

Important Concepts:

1. **Functions:** Defining and using parameters in functions, adhering to best practices like ensuring no side effects, favouring pure functions, and minimising global variables.
2. **Control Flow:** Implementing conditional statements within functions to control program flow and determine pricing based on user choices.
3. **Readability:** Enhancing code readability through meaningful variable names, clear comments, proper indentation, and organised structure.

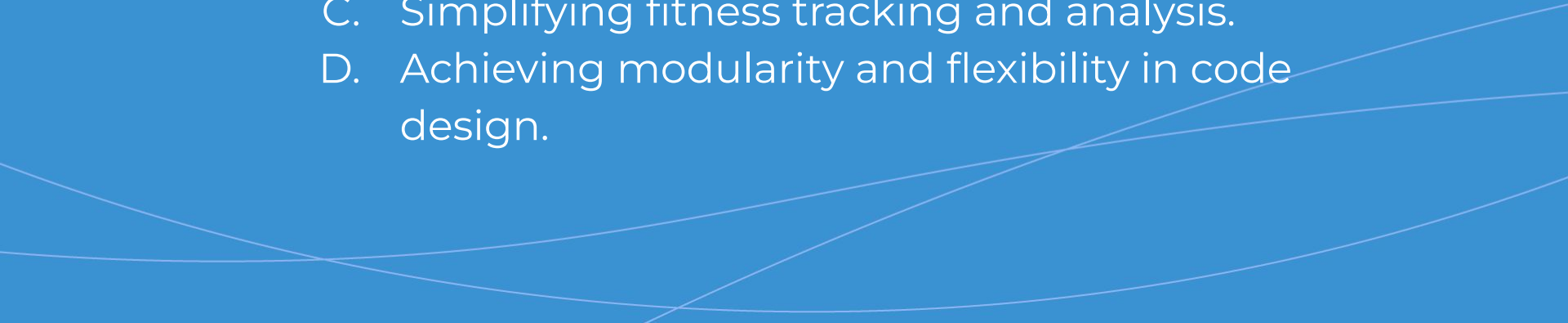
Advanced

Challenge:

- Provide optional context to the user regarding their Fitness Assessments or Health Assessments.



What is a key benefit of using higher-order functions in FitLife Gym's advanced programming scenarios?

- A. Adding more complex graphics.
 - B. Enhancing social media integration.
 - C. Simplifying fitness tracking and analysis.
 - D. Achieving modularity and flexibility in code design.
- 

Summary

User-defined Functions

- ★ User-defined functions assist in encapsulating and reusing functionality in your program/code.

Higher-order Functions

- ★ Where functions take other functions as parameters or return functions as results.



Questions and Answers

Questions around the Case Study

