



CoGrammar

DS PORTFOLIO SESSION 3



**SKILLS
FOR LIFE**

SKILLS BOOTCAMPS



Department
for Education

Data Science Session 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](#)

Data Science Session 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**

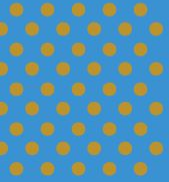
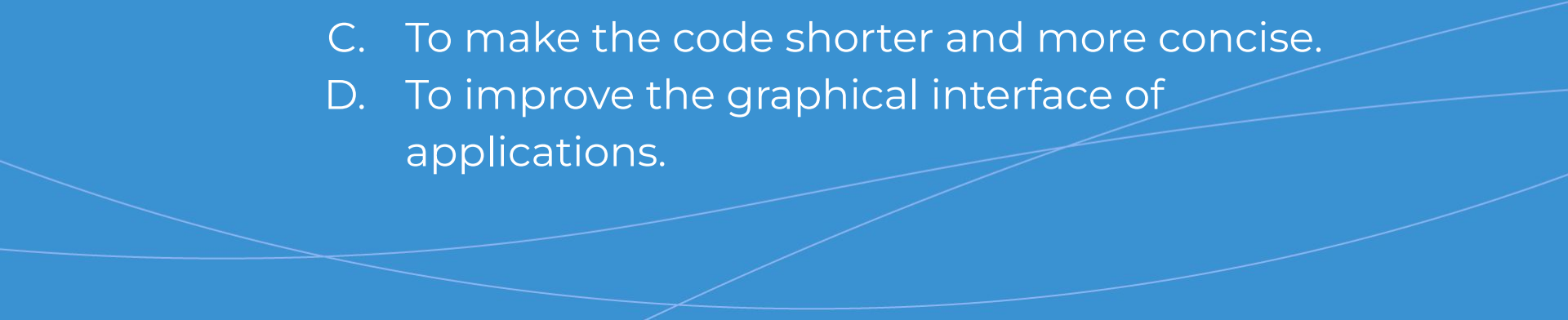
- Complete all mandatory tasks by 24th March 2024.
- Record an Invitation to Interview within 4 weeks of course completion, or by 30th March 2024.
- Achieve 112 GLH by 24th March 2024.

✓ **Criterion 4: Employability**

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





What is the primary goal of defensive programming?

- 
- A. To enhance the speed of the code.
 - B. To prevent and handle potential errors in the code.
 - C. To make the code shorter and more concise.
 - D. To improve the graphical interface of applications.
- 


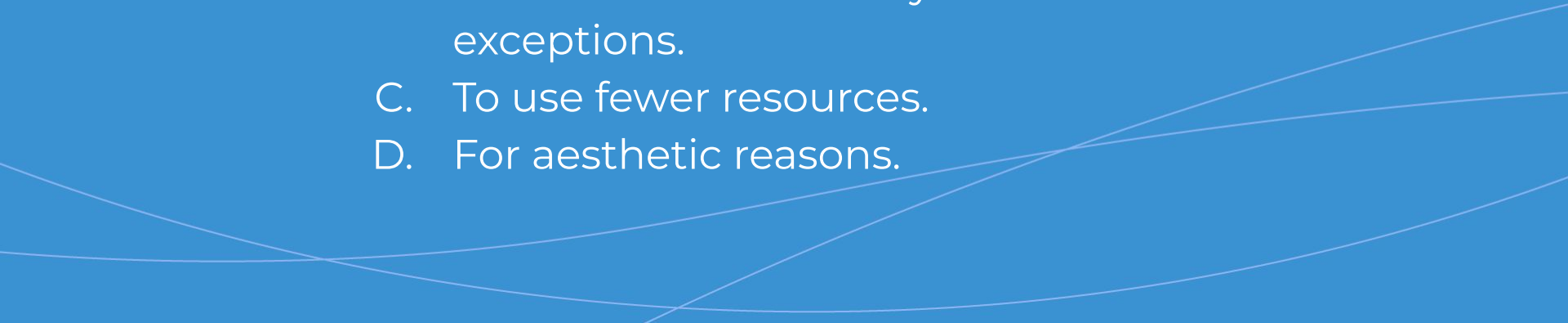


Which Python keyword is used for handling exceptions?

- 
- A. handle
 - B. error
 - C. try
 - D. catch
- 



Why would a developer create custom exceptions?

- 
- A. To make the code run faster.
 - B. To handle very specific error conditions that are not covered by standard exceptions.
 - C. To use fewer resources.
 - D. For aesthetic reasons.
- 

Recap of Week 2: Iteration

For Loops

- Used for iterating over items of a collection (like lists or strings) in the order that they appear.

While Loops

- Execute as long as a specified condition is true, useful for repeated actions until a condition changes.

Nested Loops

- A loop inside another loop, enabling the handling of multi-layered data structures.

Recap of Week 2: Iteration

Controlling Loop Execution

- **'break'**: Immediately exits the loop, typically used to exit early when a condition is met.
- **'continue'**: Skips the rest of the current loop iteration and moves to the next one, often used to skip over certain items.

```
# Example of a nested loop
for row in range(3): # Outer loop
    for col in range(3): # Inner loop
        print(f"Cell ({row}, {col})")

# Use of break and continue
for number in range(10):
    if number == 5:
        break # Exit loop
    if number % 2 == 0:
        continue # Skip even numbers
    print(number)
```

SafeBank

- **Background:** You have been tasked to revamp SafeBank's online banking interface.
- **Challenge:** Revamp the interface by ensuring user-friendliness while maintaining a robust infrastructure against errors and misuse.
- **Objective:** Upgrade the cart using loop constructs:
 - Implement input validation using If statements
 - Manage unexpected errors by gracefully by implementation of exception handling.
 - Account for potential edge cases and security loopholes.

Paws n Cart Part II

- **Programming Needs:**
 - Dynamic item addition/removal using "while" loops.
 - Efficient item display and management with "for" and nested loops.
 - Robust handling of user input and cart modifications.

Validating Email Address Format

```
# Example: Validating an email address format  
email = input("Enter your email: ")  
if "@" in email and "." in email:  
    print("Valid email address.")  
else:  
    print("Invalid email address.")
```

Demo: Input Validation Using If Statement

Here we're using if statements combined with the `isdigit()` method to validate whether the user has entered a phone number.

```
phone_number = input("Enter your phone number: ")
if len(phone_number) == 10 and phone_number.isdigit():
    print("Valid phone number.")
else:
    print("Invalid phone number.")
```

Demo: Using Try-Except to Handle Exceptions

We include the code that we would like to evaluate within the scope of the the try block and include the potential exception combined with the error message in our except block.

```
try:
    age = int(input("Enter your age: "))
    print("Age entered:", age)
except ValueError:
    print("Invalid age. Please enter a number.")
```

Types of Errors

- **Syntax Error:** Occurs when the interpreter is unable to parse the code because it violates Python language rules, such as improper indentation, incorrect keyword usage, or incorrect operator use.
- **Runtime Error:** Runtime errors are difficult to debug because they occur in real time and are difficult to reproduce.
- **Logical Error:** In Python, a logical error occurs when code runs without any syntax or runtime errors but produces incorrect results due to flawed logic.

Valentina's Shopping Cart

Your challenge is to develop a robust banking system that allows for standard banking operations while maintaining the defensive security of the program.

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

Checking Balances
Transferring Funds
Depositing Money
Withdrawing Money
Account Limits
Confirmation Prompts

Step-by-Step Tasks:

1. **User-Input:** Conceptualise how your program will receive user input, while prioritising user friendliness/experience.
2. **Input-Validation:** Consider the different ways in which users can possibly provide input and clean this data before processing.
3. **Exception Handling:** Handle predictable/common user errors by implementing the try-except blocks for the respective exceptions associated with these errors.
4. **Edge Cases:** Consider edge cases such as the possibility of a mathematical operation where division by zero can occur, and account for these.

Advanced

Challenge:

- Explore different Python GUI libraries to take your program to the next level with a graphical interface.

Summary

Input Validation

- ★ Input validation is the process of testing input received by the application for compliance against a standard defined within the application.

Exception Handling

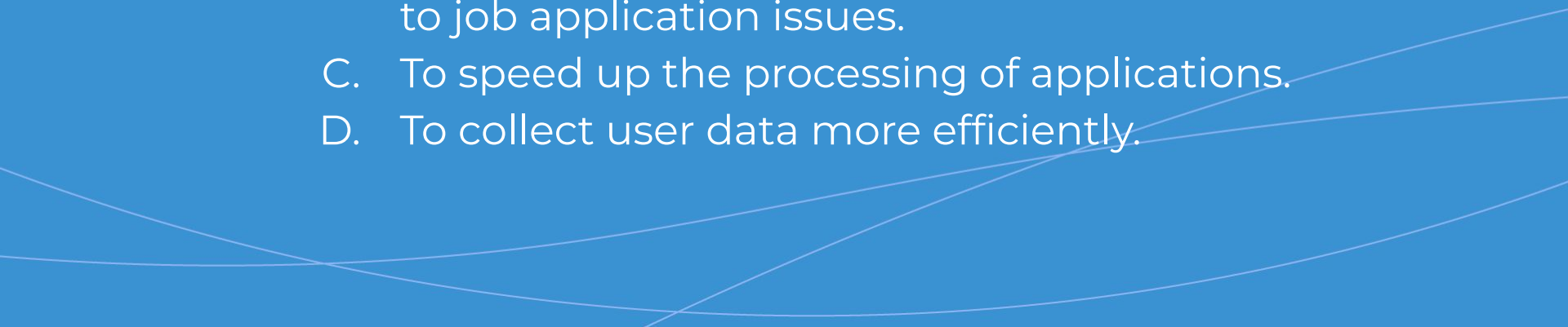
- ★ Exception handling is the process of responding to unwanted or unexpected events when a computer program runs.

Error Types

- ★ The most common types of errors that can be found while programming in Python are categorised as syntax errors, runtime errors or logical errors


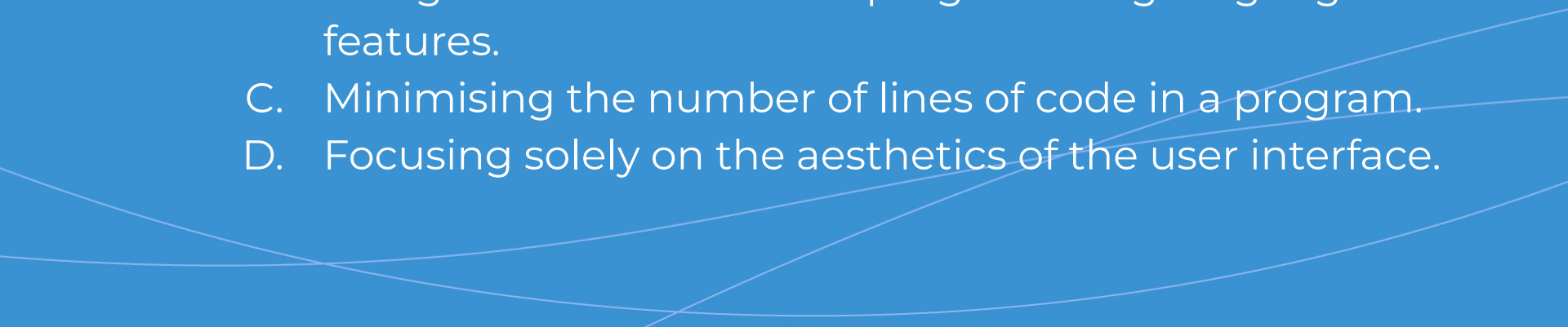


In the CareerCrafter system, why might you define a custom exception like `InvalidApplicationError`?

- A. To automatically correct user input errors.
 - B. To provide a specific error message related to job application issues.
 - C. To speed up the processing of applications.
 - D. To collect user data more efficiently.
- 



What is a key strategy of defensive programming?

- 
- A. Writing code that handles potential errors and unexpected user inputs.
 - B. Using the most advanced programming language features.
 - C. Minimising the number of lines of code in a program.
 - D. Focusing solely on the aesthetics of the user interface.
- 



Questions and Answers

Questions around the Case Study

