# GCSE OCR

Computer Science
J277

2

# Defensive design

Unit 8
Logic and languages

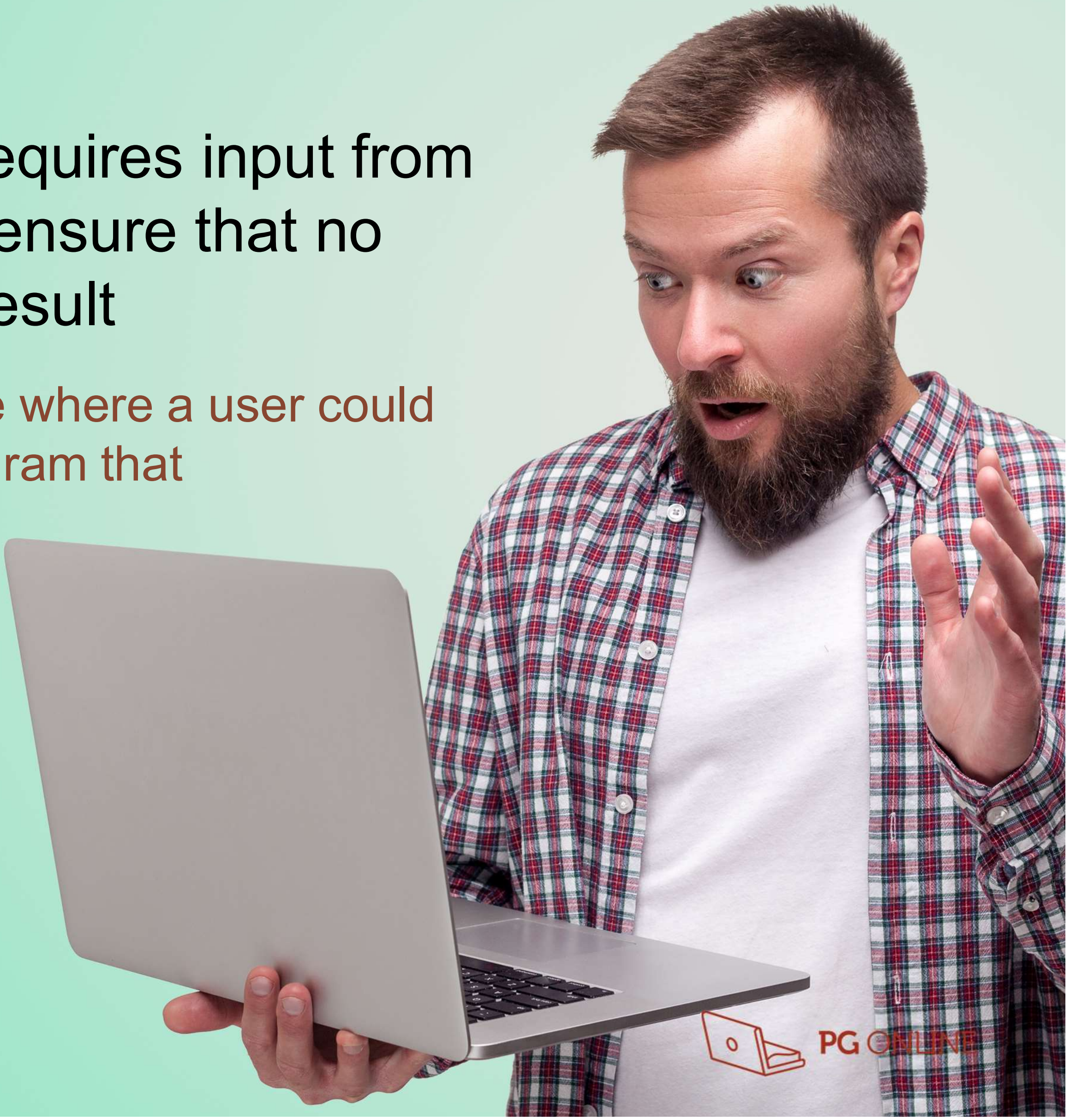
PG ONLINE

## Objectives

- Describe defensive design considerations:
  - Input validation
  - Anticipating misuse
  - Authentication
- Understand how to make maintainable programs including:
  - The use of sub programs
  - Naming conventions
  - Indentation
  - Commenting

## Starter

- When a program requires input from a user, it needs to ensure that no errors occur as a result
  - What is an example where a user could enter data to a program that causes it to crash?



## Starter

Answers

- What is an example where a user could enter data to a program that causes it to crash?

```
age = input("Please enter your age: ")
age = age + 1
```

Results in a crash as an input is a string and it is trying to do a calculation with it

## Data validation

- Data validation routines can ensure that data entered is of the right type – for example, an integer
  - Validation cannot ensure that the user has not entered a wrong value, or made a spelling mistake in a name
  - It can only ensure that the data is reasonable and conforms to a set of rules
- What other validation checks could you apply to data entered by the user?

# Types of validation check

**Answers**

| Check | Example |
|---|---|
| Range check | A number or date is within a sensible/allowed range |
| Type check | Data is of the right type, such as integer, letter or text |
| Length check | Text entered is not too long or too short – for example, a password is between 8 and 15 characters |
| Presence check | Checks that data has been entered, i.e. the field has not been left blank |
| Format check | Checks that the format of, for example, a postcode or email address is correct |

## Example

- What sort of validation check is made in this algorithm?

```
postcode = input("Please enter postcode:")
if postcode.length < 6 OR
   postcode.length > 8 then
   print("Invalid postcode")
endif
```

- Rewrite the algorithm so that the program keeps asking the user to enter a postcode until the entry is valid

## Solution

Answers

- You need a WHILE or a REPEAT loop

```
postcode = input("Please enter postcode:")
while postcode.length < 6 OR
      postcode.length > 8
    postcode = input("Invalid postcode –
                          please re-enter")
endwhile
```

- How many times will the loop be performed if the user enters IP6 4DF?

## Verification

- Validation can only check that the data entered is reasonable
- Verification is used to double-check that the data has been typed in correctly
- For example, a user setting a new password may be asked to type it in twice
    - If the two passwords don't match, they will be asked to enter the password again
    - This is known as double-entry verification

## Worksheet 2

- Now complete **Task 1** on **Worksheet 2**

## Authentication routines

- Authentication routines are used to make sure a person is who they claim to be
  - What is a common method of online authentication, for example when you log in to a website with which you have previously registered?

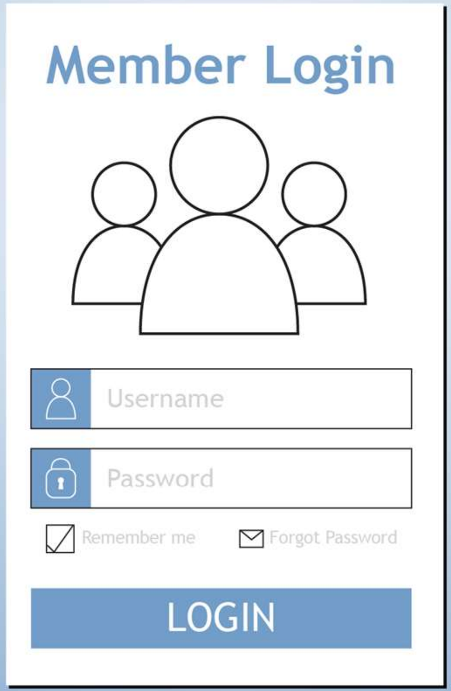
## Password routines

Answers

- Commonly, you are asked to enter a User ID and a password
  - Once you have entered the User ID, the website looks up your password in a database
- If the user ID cannot be found, an error message is displayed
  - What happens if you enter the wrong password?

## Entering a password

- You usually get three attempts to get your password, and then you will be locked out



## Anticipating misuse

- Why are you often only allowed a finite number of tries before being locked out?

## Three tries and you're out!

- It may be that you have forgotten your password, and you need to be given a reminder, so three tries is enough for the average user
- BUT a hacker may be trying out dozens of likely passwords to try and get the correct one
    - There are software programs which will try out every combination of letters, numbers and special characters – this is known as a brute-force attack
    - Use a password of 8 characters or more to make it more difficult to hack!

## Worksheet 2

- Now complete **Task 2** on **Worksheet 2**

## Maintainable programs

- Programs need to be maintained
  - This will be to improve the code, fix bugs or add new features to the program
  - It may be carried out by the original programmer or different programmers
- It is important that programs are written in a way to make them easily maintainable. This includes:
  - The use of sub programs (functions and procedures)
  - Using appropriate naming conventions
  - Indentation
  - Commenting

## Using sub programs

- Sub programs include functions and procedures
    - Well written sub programs will take inputs (through parameters) and if necessary return a value
    - They should be written so that they can be reused multiple times in the program or by other programs
    - The two programs below are for a function that works out the area of a circle. Which is more easy to reuse and will help create maintainable code?

```
function circle(radius)
  area = 3.14 * radius^2
return area
```

```
function circleFive()
  area = 3.14 * 5^2
return area
```

## Using sub programs

**Answers**

- The program on the left takes any sized radius as an input
  - This means that it is reusable many times in the program and in other programs
  - This will make a larger program easier to maintain as there will be just one function to calculate the area of a circle

```
function circle(radius)
  area = 3.14 * radius^2
return area
```

✓

```
function circleFive()
  area = 3.14 * 5^2
return area
```

✗

## Naming conventions

- The two programs below are the same algorithm
  - Which is easier to understand? Why?

```
a = float(input())
b = float(input())
c = float(input())
d = a + b + c
e = d / 3
print(e)
```

```
num1 = float(input())
num2 = float(input())
num3 = float(input())
total = num1 + num2 + num3
average = total / 3
print(average)
```

# Naming conventions

Answers

- The two programs below are the same algorithm
  - Which is easier to understand? Why?

```
a = float(input())
b = float(input())
c = float(input())
d = a + b + c
e = d / 3
print(e)
```

✗

```
num1 = float(input())
num2 = float(input())
num3 = float(input())
total = num1 + num2 + num3
average = total / 3
print(average)
```

✓

- It is important to use meaningful names for variables, constants, functions and procedures
  - This makes code easier to read and understand

## Indentation

- Look at the following pseudocode:

```
tables = 12
rows = 12
for i = 1 to tables
for j = 1 to rows
answer = i * j
next j
next i
print(str(i) + "x" + str(j) + " = " + str(answer))
```

- Why would the use of indentation improve it?
    - Why may indentation be essential in some languages?

# Indentation

Answers

```
tables = 12
rows = 12
for i = 1 to tables
    for j = 1 to rows
         answer = i * j
    next j
next i
print(str(i) + "x" + str(j) + " = " + str(answer))
```

- Why would the use of indentation improve it?
  - Indentation makes it possible to easily see which lines of code are part of different structures
- Why may indentation be essential in some languages?
  - Some languages use braces { } to show where structures start and end, but some, such as Python use indentation

## Commenting

- Comments in code help other programmers to understand your code
  - They also help you understand your code when you go back to it at a later time
- Which parts of programming code tend to be commented?
  - Which parts are typically not commented?

//

## Commenting

Answers

- Comments are usually written for:
  - Parts of a program/algorithm that are difficult to understand
  - At the start of a function or procedure to explain what it does
- Comments usually aren’t written for:
  - Every line of code
  - To explain parts of code that are obvious
  - To explain syntax used in the programming language – programmers are expected to look up and learn parts of the language that they don’t understand

## Plenary

- Work in pairs to explain the following terms and how they can improve the design of programs
  - Input validation
  - Anticipating misuse
  - Authentication
  - Use of sub programs
  - Naming conventions
  - Indentation
  - Commenting

## Plenary

Answers

- Input validation – checking input meets certain rules, e.g. the type of data
- Anticipating misuse – preventing too many entries of a password to make it harder for hackers to guess
- Authentication – entering data twice or checking from an alternative source
- Use of sub programs – creates reusable code where bugs can easily be fixed
- Naming conventions – good use of variables and sub program names makes programs easier to read
- Indentation – makes programs easier to read
- Commenting – helps programmers understand what a program does and how it does it

PG ONLINE