

T<sub>eaching</sub> L<sub>ondon</sub> C<sub>omputing</sub>

# Programming for GCSE

## Topic 4.2: Faults and Debugging



**COMPUTING AT SCHOOL**  
EDUCATE · ENGAGE · ENCOURAGE



SUPPORTED BY  
**MAYOR OF LONDON**



# Aims

- What goes wrong and when?
    - Understanding the faults at different stages of program execution
  - Techniques for debugging
    - Form a hypothesis
    - Halve the problem
  - Debugger
    - Principles
    - Demo
-

# Teaching Issue

- Simple errors can be disproportionately difficult at first
    - Does it help to understand more?
  - Teach some tactics for fixing errors
    - The limits of 'just changing something'
  - When to introduce the debugger
    - Is it useful?
-



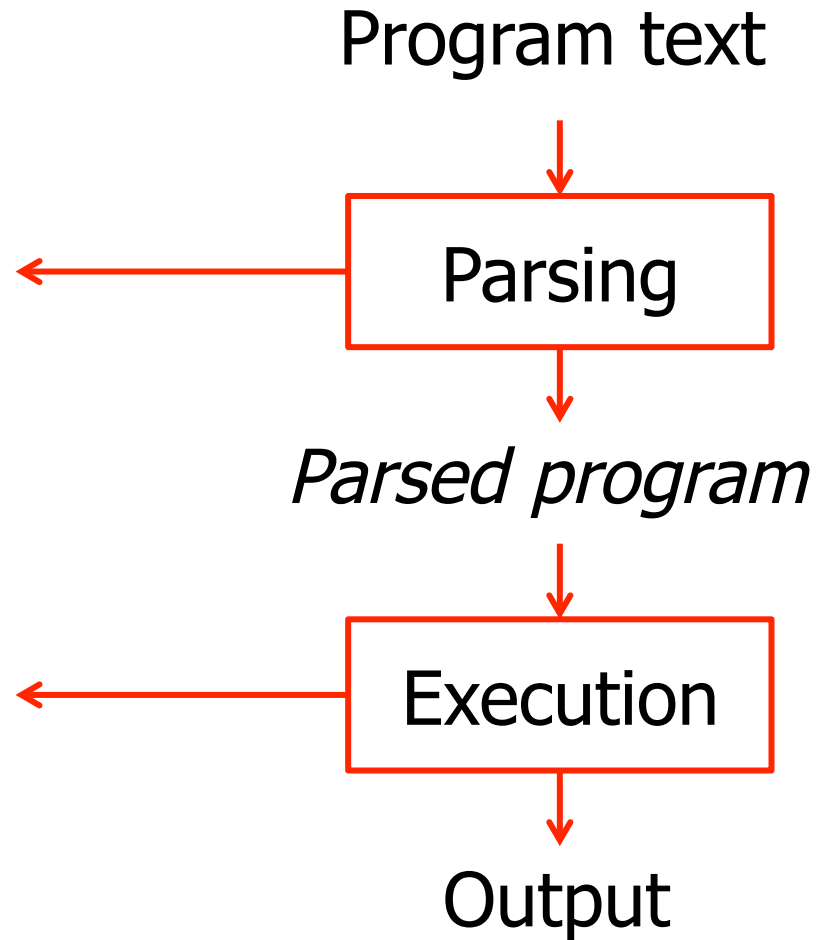
# UNDERSTANDING WHAT GOES WRONG

... and when

---

# Types of Errors

- Syntax errors
  - Brackets
  - String
  - Indentation
- Execution errors
  - Names
  - Expressions
  - Variables



# Types of Errors

- Syntax
    - *The words make no sense*
    - *The words do not make a sentence*
  - Execution errors
    - *I cannot understand that sentence*
    - *I know what you mean, but I cannot do that*
  - Python has more execution errors
    - Other languages have a type checker
-

# Bugs

- The program works, but it isn't the program you wanted!

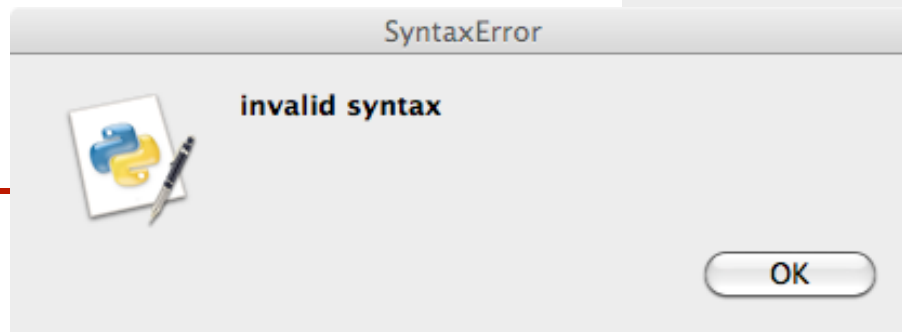
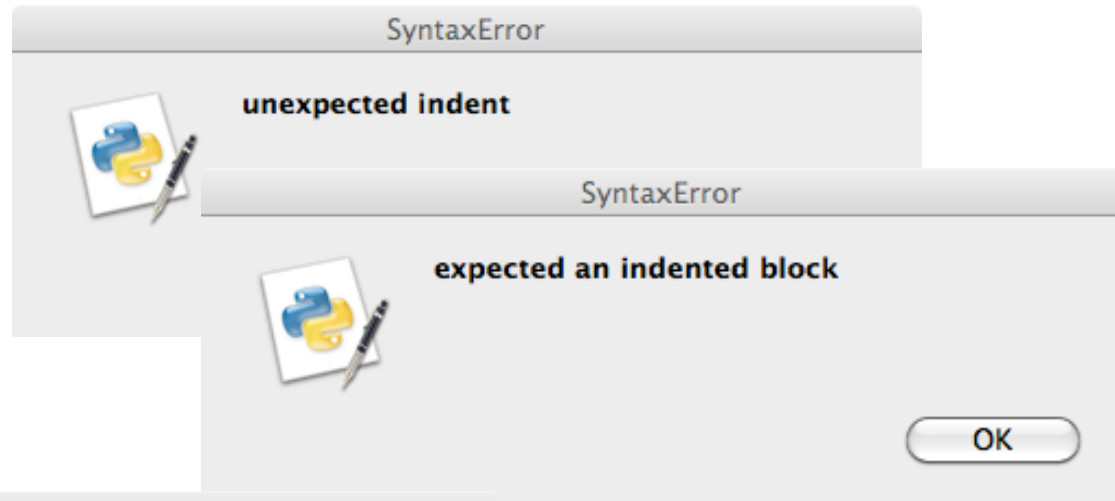
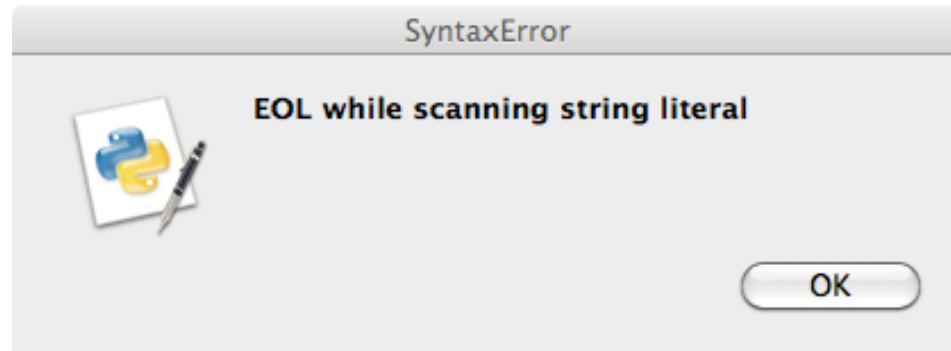
```
number=int(input("Enter a Binary Number> "), 3)
print("In Base 10, the number is: ", str(number))
print("In Octal, the number is: ", hex(number))
print("In Hexadecimal, the number is: ", oct(number))
```

```
Enter a Binary Number> 1011
In Base 10, the number is:  31
In Octal, the number is:   0x1f
In Hexadecimal, the number is:  0o37
```

---

# Syntax Errors

- Strings: must close
- Indentation
  - Proper boxes
- Brackets
  - Must match

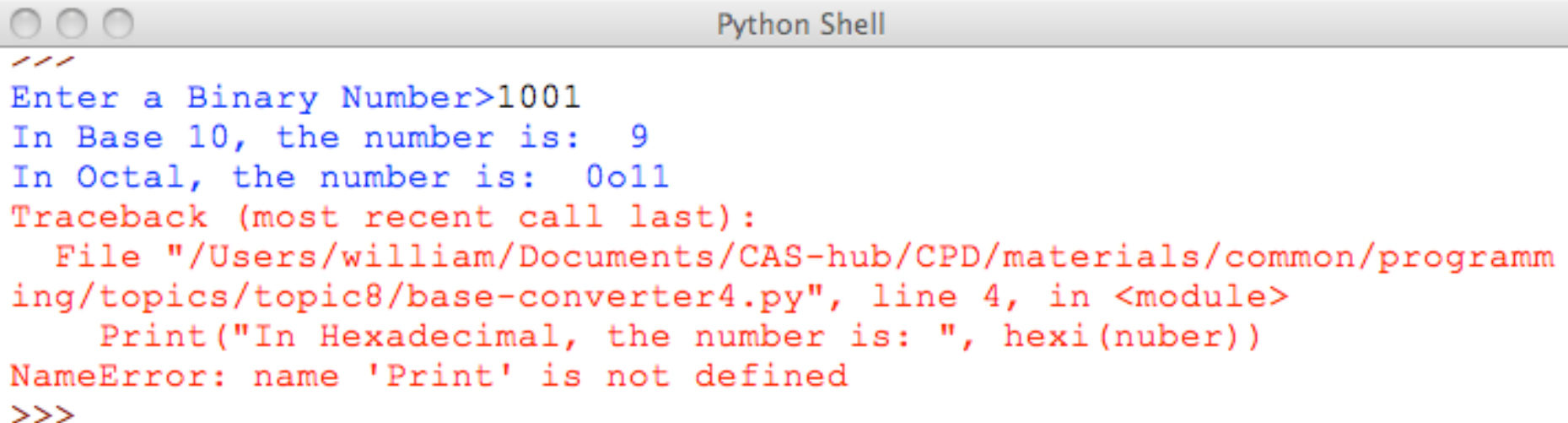




# Execution Errors – Names

- Names: As variables are not declared, any name could be a variable

```
number=int(input("Enter a Binary Number>" ), 2)
print("In Base 10, the number is: ", str(number))
print("In Octal, the number is: ", oct(number))
Print("In Hexadecimal, the number is: ", hexi(nuber))
```



A screenshot of a Python Shell window. The window has a title bar with three window control buttons (red, yellow, green) on the left and the text "Python Shell" in the center. The main area of the window contains a text-based interaction. It starts with a prompt "Enter a Binary Number>" followed by the input "1001". The next two lines show the output of the program: "In Base 10, the number is: 9" and "In Octal, the number is: 0o11". The fourth line shows a red error message: "Traceback (most recent call last):", followed by the file path and line number: "File \"/>Python Shell

```
Enter a Binary Number>1001
In Base 10, the number is: 9
In Octal, the number is: 0o11
Traceback (most recent call last):
  File "/Users/william/Documents/CAS-hub/CPD/materials/common/programm
ing/topics/topic8/base-converter4.py", line 4, in <module>
    Print("In Hexadecimal, the number is: ", hexi(nuber))
NameError: name 'Print' is not defined
>>>
```

# Execution Errors

- Names
    - As variables are not declared, any name could be a variable
  - Variable
    - Must be assigned before used
  - Expression
    - The operator does not exist
    - E.g. `"David" — "Cameron"`
    - The operator has no answer for the values
    - E.g. `"David" [ 5 ]`
-



# DEMONSTRATIONS

---

# Demo 1

base-converter4.py - /Users/william/Documents/CAS-hub/CPD/materials

```
number=int(input("Enter a Binary Number> "), 2)
print("In Base 10, the number is: ", str(number))
print("In Octal, the number is: ", oct(number))
Print("In Hexadecimal, the number is: ", hexi(nuber))
```

---

# Demo 2

letters-e2.py - /Users/william/Documents/CAS-hub/CPD/materials/comm...

```
print("Enter your name a letter at a time")
print("End with a full stop")

complete = False

#Loop inputting each letter until the name complete
while not complete :
    letter = input("Next letter: ")
    if letter == '.'
        complete = True
    el :
        name = name - leter

print("Your name is", name)
```

# TACTICS FOR FINDING ERRORS

Test often

---

# Where is the Error?

- Hard to find if you are looking in the wrong place
    - Syntax errors may 'appear' after real location
  - Use comments to shrink program
  - Look at the line number in the message
  - Print something before error
-

# Importance of 'Hypothesis'

- "I think the problem is ..."
  - Have an idea .. test it ... revise it
  - Alternative is to make 'random' changes
-



# DEBUGGER

Watch the program working

---

# Debugger - Principles

- Breakpoint
    - Stop the program in progress
  - Observe the values of variable
-

# DEMO OF DEBUGGER

See separate demo.

---

# Summary

- Finding errors is difficult
    - Lots of errors at first
  - Does it help to understand the types of errors
  - Teach good habits
    - Be systematic in finding errors
    - Have a 'hypothesis'
  - Bad habits: make changes without thought
-