Objective 11:

How to handle exceptions for validation

In this objective you learn how to prevent your program crashing if it receives invalid data.

Tasks

1. Try entering the following program and use the test table of data to see what happens:

```
#Run time errors
num1 = float(input("Enter a number to divide: "))
num2 = float(input("Enter a number to divide by: "))
print(num1, "divided by",num2, "is",num1/num2)
```

Test	First number	Second number	Expected
1	6	2	3.0
2	8	7	1.142
3	4	0	Crash – divide by zero
4	5	b	Crash – invalid character entered

A program should not crash because it has bad data input. The potential for errors should be considered by the programmer.

2. Try entering the following program and use the test table of data to see what happens:

```
#Run time errors
try:
      #Enter two numbers, trapping errors
      num1 = float(input("Enter a number to divide: "))
      num2 = float(input("Enter a number to divide by: "))
      try:
            #Calculate division trapping division by zero
            answer = num1/num2
            print(num1, "divided by",num2, "is",answer)
      except:
            print(num1, "divided by",num2,"is infinity.")
#output error message if a string was entered
except:
      print("Invalid data. Unable to continue.")
```

Objective 11: Key learning points

How to handle exceptions for validation

- Exception handling code must be indented.
- There are 3 types of errors in programming:
    - **Syntax errors** when you mistype a keyword, the program doesn't recognise it as a valid command and will not run.
    - **Logic errors** where the program runs but you get the wrong result, perhaps not always consistently! Logic errors are called **bugs**.
    - **Run-time errors** that occur in exceptional circumstances such as division by zero and incorrect data type input. These should be trapped with **exception handling** commands.
- Fixing errors in code is known as **debugging**.
- Preventing invalid data input is known as **validation**. There are a number of different types of validation including:
    - Presence checks: did the user actually enter any data?
    - Range checks: does the input fall within a valid range of numbers?
    - Format check: does the input match a recognised format, e.g. LLNN NLL for postcode
    - Length check: is the input the required length of characters, e.g. minimum 8 character password.
- Most validation checks can be performed with selection and iteration statements, but sometimes an error could occur if one data type is converted to another, e.g. a string to an integer if the character is not a number.

Objective 11: Key words

Try… except

Example code:

```
try:
        …

except:
        …
```

'Try' is the keyword to put before a run time error is likely to occur. The code following 'except' is the code to execute if an error occurs.