Python programming companion

Pedagogy

Back in the 1980's I have fond memories of typing in programs from computer magazines, to see what they would do. Inevitably, at 10 years old, I wasn't very accurate copying the listings and there were errors. I had to study the program carefully, and correct my mistakes.

The programs would eventually run, and I would be delighted that the computer did something it couldn't do before. The programs were very simple, so I began experimenting with changing a few variables (not that I understood what they were back in the day) to see what would happen. This intrigued me, and having picked up the "Programmers guide" and "Programming keywords" for Locomotive Basic that had come with my computer, I began to experiment with adding in extra little bits to the code, to see what would happen. Gradually I became more confident. I began to want more knowledge of keywords, to solve a particular problem I had. Using the programming keywords as a handy reference I learned the commands I could use, their syntax, and I gradually memorised them. Programming was challenging and fun.

As I became more confident I began setting myself bigger challenges, writing code from scratch, and learning more through the process. I began to learn of more appropriate techniques, and my programs became more sophisticated. As I learned new languages, I would begin to read programming books, but skip much of the text to look at the code examples, transitioning my knowledge of keywords from one language to another.

It is this learning journey to becoming a confident programmer I want to share with my students:

- Typing up code, seeing what happens: the confidence from not starting from a blank screen.
- Modifying the code to do something different: beginning to understand what the code means.
- Attempting challenges using only the commands learned: applying new knowledge gained.
- Beginning to write bigger programs with less support: becoming independent.

This guide facilitates that journey in Python: a good starting point for the basics of algorithms before getting into event driven and object-oriented approaches. However, this is not the end. The guide will continue to be developed with an increasing number of challenges. The aim being for students not to solve every challenge, but to choose for themselves: a personalised approach to a common goal.

There is a console Visual Basic version of this guide too.

Codes used in this companion

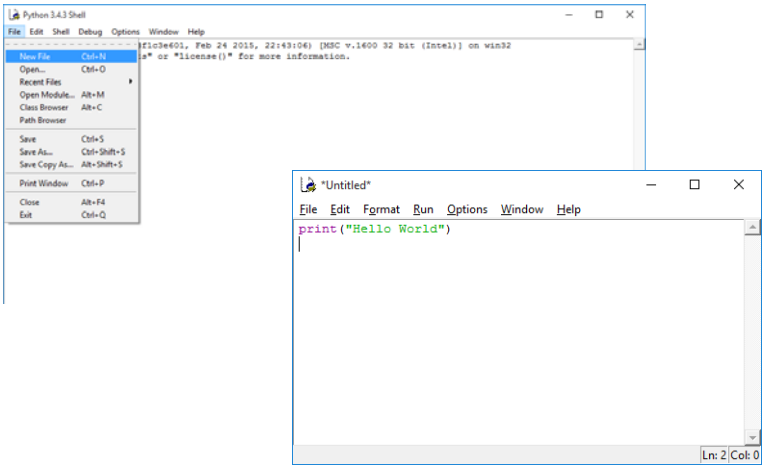
In the code, a continuation of the same line is indicated with an arrow symbol: →
The difficulty of a challenge is graded by 🛠 icons. Students can choose what challenges to complete depending on their confidence. 'G' indicates that it is probably suitable for GCSE, 'A' indicates that the challenge is suitable for A'level. The expectation is that students will complete the challenges in two languages, one at GCSE, and one at A'level.

Enjoy!

David Hillyard

Getting started in Python

1. Open IDLE (Python). This guide is written for version 3.4
2. Choose File… New File…



3. A new window will open into which you write your code.
4. The program must be saved before it will run.
5. To run your program press F5.
6. When you save your program, make sure to add .py extension to the filename. This will not only make it easier to find your programs, but will also ensure the text formatting is not lost in the editor.

Please take screen shots of your code and paste it in the Starter Pack Check list. You must bring the completed checklist with you on your first day

Objective 1:

Understand how to output text strings

In this objective you learn how to output text to the screen.

Tasks

1. Try entering the following commands and see what happens:

```
print("Hello World")
```

2. Try entering the text without the speech marks and see what happens:

```
print(Hello World)
```

Note the difference. One works and one doesn't! If you want to output text to the screen it must be included in speech marks. Text is called a string in programming and must be enclosed in double quotes. Python is a case sensitive language, the commands, such as 'print' must be entered in lowercase.

3. Try entering the following commands and see what happens:

```
print("Hello World","this is my first program.")
```

Note how a comma joins strings together. This is called concatenation. In Python a comma adds a space between the strings. If you don't want this, use a + instead of a comma.

4. Try this sequence of instructions:

```
#Start of message
print("Hello World","this is my first program.")
#Blank line
print()
#End of message
print("I am learning to code...")
print("...and it is fun")
```

Note the hash symbol enables you to insert a comment in the code. The computer ignores the comments, but they help to introduce the program and allow you to leave notes in the code to make it easier to understand later.

5. Change the program so it outputs this instead:

```
Computers only do exactly as they are told…
…so you need the code to be correct!

If you make any mistake with the commands, it won't work
```

Objective 1: Key learning points

Understand how to output text strings

- Text is known as a **string** in programming and must be enclosed in double quotes.
- Strings can be joined together using a comma or plus symbol. This is known as **concatenation**.
- When a program is run it is called **executing** the program.
- A set of instructions that execute one after another is known as a **sequence**.
- Comments can be inserted into the code using a hash symbol. These are helpful to make notes within the program so that the code can be understood again at a later date or by another programmer.

Objective 1: Key words

print()

Example code: `print(x)`

Purpose: to output x to the screen followed by a carriage return.

To disable the carriage return, use: `print(x,end='')`

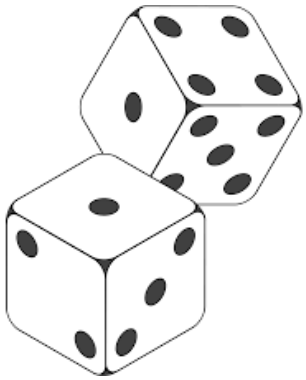
Objective 1: Challenges

Visual dice challenge

Difficulty: 🛠 G

Write a program that will output the number 5 on a dice like this:

```
ooooooooooo
o         o
o   #     o
o     #   o
o   #     o
o         o
ooooooooooo
```



ASCII art challenge

Difficulty: 🛠 G

ASCII *(pronounced: as-key)* art is a graphic design technique of making pictures using only the 95 printable characters on a keyboard.

Write a program that outputs your name in ASCII Art.

You may find it easier to create the final output in Notepad first and then copy the lines into the programming language.

Objective 2:

Understand how to input strings and numbers into variables

In this objective you learn how to get input from the keyboard to use in your program.

Tasks

1. Try entering the following commands and see what happens:

```
#Inputting strings in Python
print("Hello")
name_entered = input("What is your name? ")
print("Thank you",name_entered)
```

2. Try entering the following commands and see what happens:

```
year = int(input("What year is it please? "))
print("Ah, it is",year,"thank you.")
```

3. Change the program so it asks you for your name and your age, outputting for example:

```
Thank you Dave. You have registered an age of 15.
```

Objective 2: Key learning points

How to input strings and numbers into variables

- Data is input by a user into a **variable**.
- Variables have a data type: string, integer or float as examples, indicating how much memory they will use and the type of data they will store.
- Python does not require variables to be declared before they can be used.

Objective 2: Key words

input

Example code: `x = input("Enter your name:")`

Purpose: to store text input at the keyboard into a variable, x which can be used later in the program without inputting again.

int

Example code: `x = int(x)`

Purpose: convert variable x to an integer. Most useful to convert a string input to a number because the character "5" is not the same as the number 5 to a computer.

Combining input and int enables the input of a number. E.g.

```
x = int(input("Enter your age:"))
```

float

Example code: `x = float(x)`

Purpose: convert variable x to a floating point (decimal) number. Most useful to convert a string input to a number with decimal places because the characters "5.5" are not the same as the number 5.5 to a computer.

Objective 2: Challenges

Simple adder challenge

Difficulty: 🛠 G

Write a program that asks the user for two numbers, adds them together and outputs for example:
You entered numbers 5 and 12
They add up to 17

Test marks challenge

Difficulty: 🛠 G

Write a program that will ask the user to enter three test marks out of 100 and output the average.

Temperature converter challenge

Difficulty: 🛠 G

Write a program to enter a temperature in degrees Fahrenheit and display the equivalent temperature in degrees Centigrade.
The formula for conversion is Centigrade = (Fahrenheit – 32) * (5/9)

Height & weight challenge

Difficulty: 🛠🛠 G

Write a program to convert a person's height in inches into centimetres and their weight in stones into kilograms. [1 inch = 2.54 cm and 1 stone = 6.364 kg]

Toy cars challenge

Difficulty: 🛠🛠 G

A worker gets paid £9/hour plus £0.60 for every toy car they make in a factory. Write a program that allows the worker to enter the number of hours they have worked and the number of trains they have made. The program should output their wages for the day.

Fish tank volume challenge

Difficulty: 🛠🛠 G

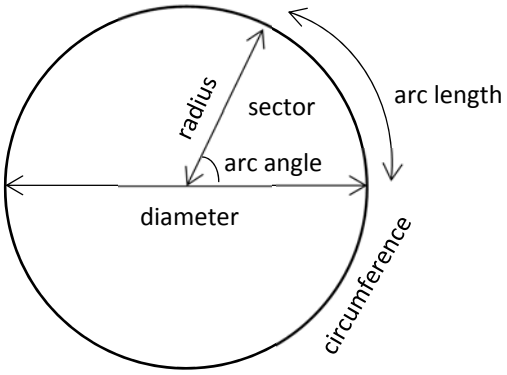
Write a program that will ask the user to enter the length, depth and height of a fish tank in cm. Calculate the volume of water required to fill the tank and display this volume in litres and UK gallons.

To calculate volume in litres, multiply length by depth by height and divide by 1000.

Circle properties challenge

Difficulty: 🛠🛠 G

Consider this circle:



Write a program that:

- Asks the user to enter the diameter of a circle.
- Outputs the radius of the circle (diameter divided by 2)
- Outputs the area of the circle (3.14 multiplied by the radius squared)
- Outputs the circumference of the circle (3.14 multiplied by the diameter)
- Asks the user to enter the arc angle
- Outputs the arc length (circumference multiplied by the arc angle, divided by 360)

Objective 3:

Understand string manipulation functions

In this objective you learn how to extract and change data that has been input.

Tasks

1. Try entering the following commands and see what happens:

```
#Working with strings
forename=input("Enter your surname: ")
forename_uppercase=forename.upper()
print("Your name in capital letters is:",forename_uppercase)
```

2. Change the program to ask for an email address, outputting the string in lowercase using `.lower()` instead of `.upper()`.

3. Try entering the following commands and see what happens:

```
#Len returns the number of characters in a string
surname = input("Enter your surname: ")
length_name = len(surname)
print("There are",length_name,"letters in your name.")
```

4. Try entering the following commands and see what happens:

```
#[:?] returns a number of characters to the left of a string
sentence = "I saw a wolf in the forest. A lonely wolf."
characters = sentence[:5]
print(characters)
```

5. Change the program to output the last 12 characters in the sentence using `[-12:]` instead of `[:5]`.

6. Try entering the following commands and see what happens:

```
#[start:end] returns a number of characters in the middle of a string
sentence = "I saw a wolf in the forest. A lonely wolf."
characters = sentence[20:26]
print(characters)
```

7. Try entering the following commands and see what happens:

```
#find returns the location of one string inside another
sentence = "I saw a wolf in the forest. A lonely wolf."
print(sentence)
word = input("Enter the word to find: ")
position = sentence.find(word)
print("The word",word,"is at character",position)
```

Objective 3: Key learning points

String manipulation functions

- Strings can be manipulated using built in functions and methods to extract characters from the left, right or middle of a string.
- You can find if one string exists inside another string.
- A built in function takes data to use in parenthesis (brackets), called a **parameter** and returns a result. E.g. `int(parameter)`
- A method (signified with a dot) applies an operation to itself. E.g. `"Hello".upper()`

Note, it is possible for functions and methods to return their value to another command rather than a variable. For example: `print("Hello World".find("World"))` works because the resultant value from find becomes the parameter for print.

It is common in programming to use as few variables as necessary in order to conserve memory. This makes the program more efficient, but often more difficult to understand.

Objective 3: Key words

.upper()

Example code: `x = y.upper()`

Purpose: To turn a string into uppercase.

x is the name of the variable to return the result to. y is the original variable.

.lower()

Example code: `x = y.lower()`

Purpose: To turn a string into lowercase.

len()

Example code: `x = Len(y)`

Purpose: To return the number of characters in a string.

x becomes the number of characters in y.

[:?]

Example code: `x = y[:z]`

Purpose: To return characters to the left of a string.

x becomes the characters. y is the string to extract characters from. z is the number of characters to extract from the left.

[-?:]

Example code: `x = y[-z:]`

Purpose: To return characters to the right of a string.

x becomes the characters. y is the string to extract characters from. z is the number of characters to extract from the right.

[?:?]

Example code: `x = y[w:z]`

Purpose: To extract characters from the middle of a string.

x becomes the middle characters of y, starting from position w, up to z number of characters.

.find()

Example code: `x = y.find(z)`

Purpose: To find the position of substring z in y.

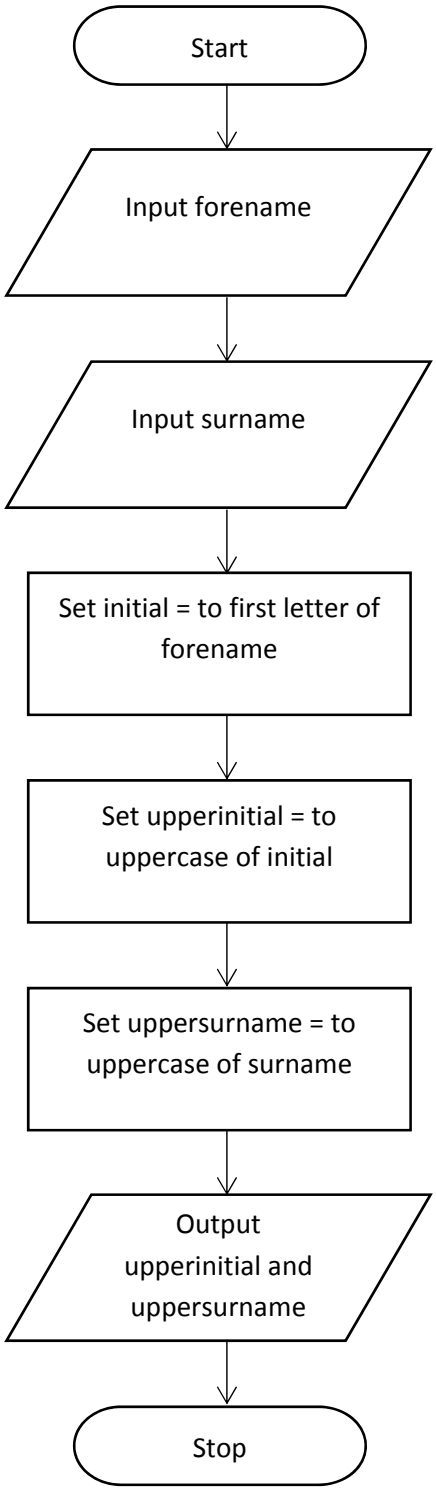
X becomes the position in the string y where z can be found. E.g. x = “Hello World”.find(”World”) returns 6 as the letter W is the sixth character in the string (starting at zero).

Objective 3: Challenges

Initials & surname challenge

Difficulty: 🛠 G

Write the program from the flowchart below to ask for a person's forename and surname, outputting their first initial and the rest of their name in capitals. E.g. ben white = B WHITE



Airline ticket challenge

Difficulty: 🛠🛠 G

Write a program that allows the user to input the name of two cities. The program should then output the first 4 characters of each city in capital letters, separated by a dash. For example, London & Madrid would be: LOND-MADR

Name separator challenge

Difficulty: 🛠🛠🛠 A

Write a program that allows the user to enter their full name on one line. The program should then output for example:

```
Forename: Simon
Surname: Hall
```

The program needs to find the space and then extract the characters to the left and right of the space into two different variables.

Word extractor challenge

Difficulty: 🛠🛠🛠🛠 A

Write a program that outputs the sentence: Quick brown fox jumps over the lazy dog. The user can then enter the word to be cut from the sentence. The sentence is then output with the word removed.

Objective 4:

Understand how to use selection statements

In this objective you learn how to make different parts of your program run depending on the value of variables.

Tasks

1. Try entering the following commands and see what happens:

```
#Using selection statements to check variables
#Ask user for the number
number = int(input("Enter a number 1-3:"))
#Check the value of the number
if number == 1: print("Number one")
if number == 2: print("Number two")
if number == 3: print("Number three")
```

   == allows you to check if a variable equals a value.

2. Try entering the following commands and run the program three times with the test data: 6, 51 and 70.

   Pay careful attention to the indentation with tabs, these are not spaces.

```
#Works out whether a candidate passed or failed
score = int(input("Enter score:"))
if score > 40:
    print("You passed")
else:
    print("You failed")
```

TAB → (before `print("You passed")`)

TAB → (before `print("You failed")`)

   Note how the program only executes instructions if the condition is true i.e. the score is more than 40, and how it is possible to use an else statement to run an alternative set of commands if the condition is false.

3. Change the program so that you have to score over 50 to pass.

4. Try entering the following commands and run the program three times with the test data:
   6 & 4, 51 & 51.

```
#Returns whether two numbers are the same
num1 = int(input("Enter a number: "))
num2 = int(input("Enter a number: "))
if num1 != num2:
      print("The numbers are not the same")

else:
      print("The numbers are the same")
```

Note != means does not equal.

5. Try entering the following commands and run the program three times with the test data:
   1, 3, 5.

```
#Using logic operators
choice = input("Enter a number 1-3: ")
if choice > "0" and choice < "3":
    print("Valid input")
else:
    print("Invalid input")
```

Note how 'and' can be used to check if both conditions are true. You can also use 'or' if only one condition needs to be true and 'not' as an alternative to does not equal.

Note how the number was input as a string, not an integer, yet greater than and less than operators still work to check if the input was between 1 and 2. That's because the ASCII value of the character is being checked, not the actual number. This approach is helpful as it prevents the program crashing if a character is input instead of a number.

6. Try entering the following commands and see what happens:

```
#Using case selection
#Ask user for the number
print("1. Add numbers")
print("2. Subtract numbers")
print("3. Quit")

choice=input("Enter your choice: ")
#Multiple branches depending on selection
if choice == "1":
    print("Add numbers chosen")
elif choice == "2":
    print("Subtract numbers chosen")
elif choice == "3":
    print("Quit chosen")
```

Note how it is possible to use `elif` to have multiple branches rather than just true or false.

Objective 4: Key learning points

How to use selection statements

- Checking the value of a variable and executing instructions depending on the outcome of the check is known as a **selection**.
- The instructions that are executed as a result of a selection is known as a **program branch**.
- The logical checking of a variable against another value is known as a **condition**.
- Elif commands can be used instead of multiple if statements.
- Code for each program branch must be indented.
- It is good practice to comment a selection to explain its purpose.
- One 'If' can be placed inside another 'If', this is known as **nesting**.

Objective 4: Key words

If… else: …

Example code:
```
If (x>0 and x<5) or x=10:
        …
else:
        …
```

x is the variable being checked. Multiple variables can be checked in one condition. Use brackets to express order of priority. In the example code, x must be between 0 and 5 or equal to 10 for the condition to be true. Else is optional and is used to branch if the condition is false. All program branches must be indented.

Logical operators that can be used in conditions include:

==	equals	<	less than	and	both conditions must be true
!=	does not equal	<=	less than or equal to	or	one condition must be true
		>	greater than	not	condition is not true
		>=	greater than or equal to		

elif

Example code:
```
if x == 3:
        …
elif x < 3:
        …
elif x >= 10 and x <= 20:
        …
else:
        …
```

x is the variable being checked. elif is used as an alternative to multiple if statements.

Objective 4: Challenges

Under age challenge

Difficulty: 🛠 G

Write a program that asks for your age. If you are over 18 it outputs the message, “Over 18”, otherwise it outputs, “Under 18”.

Water temperature challenge

Difficulty: 🛠 G

Write a program that reads in the temperature of water in a container in Centigrade and displays a message stating whether the water is frozen (zero or below), boiling (100 or greater) or neither.

Vocational grade challenge

Difficulty: 🛠 G

Write a program that allows you to enter a test mark out of 100.
The program outputs “FAIL” for a score less than 40, “PASS” for a score of 40 or more, “MERIT” for a score of 60 or more and “DISTINCTION” for a score of 80 or more.

Extended visual dice challenge

Difficulty: 🛠 G

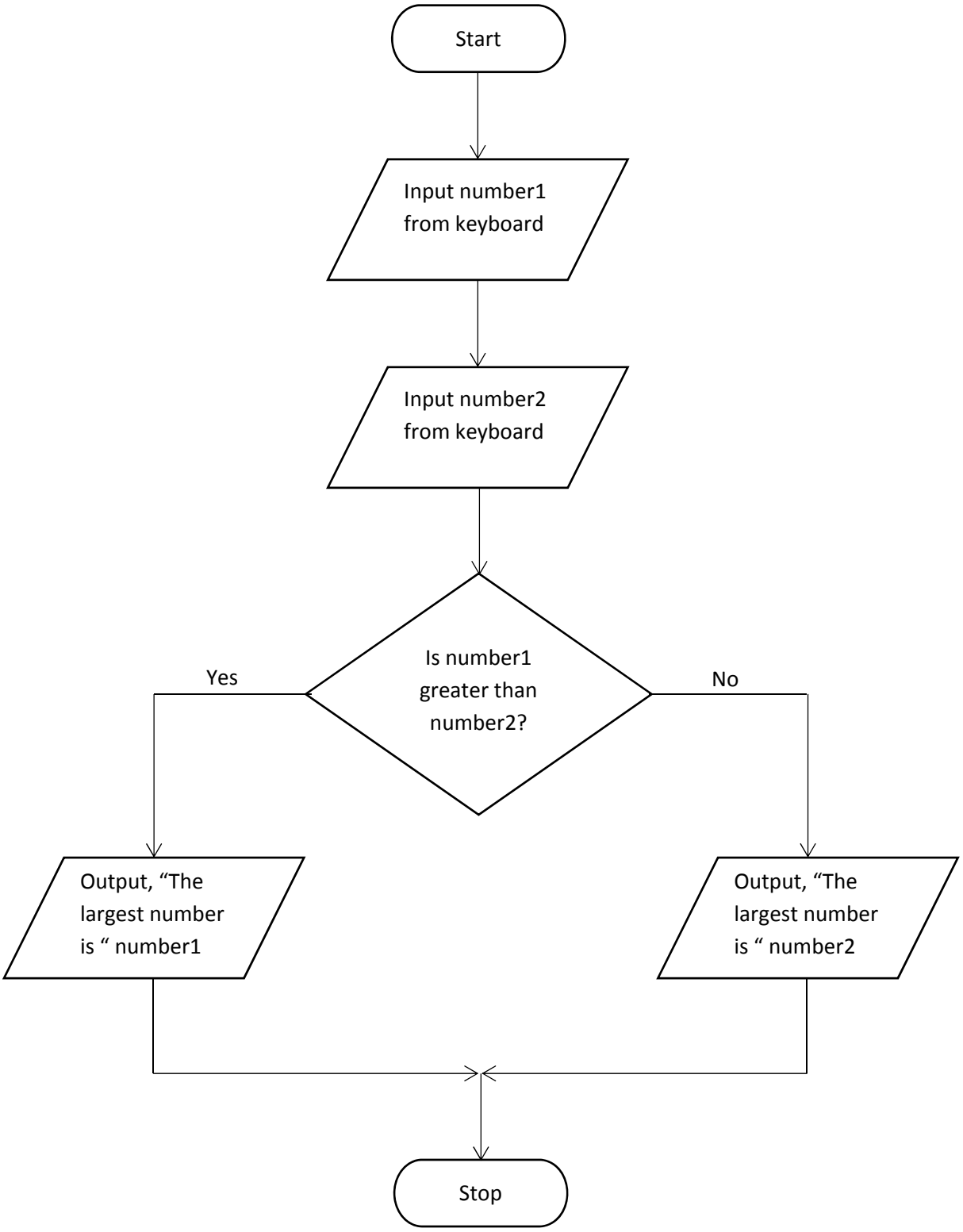
Write a program that asks for a number and outputs that number as a graphical dice. E.g.

```
oooooooooooo
o          o
o  #       o
o     #    o
o        # o
o          o
oooooooooooo
```

Greatest number challenge

Difficulty: 🛠 G

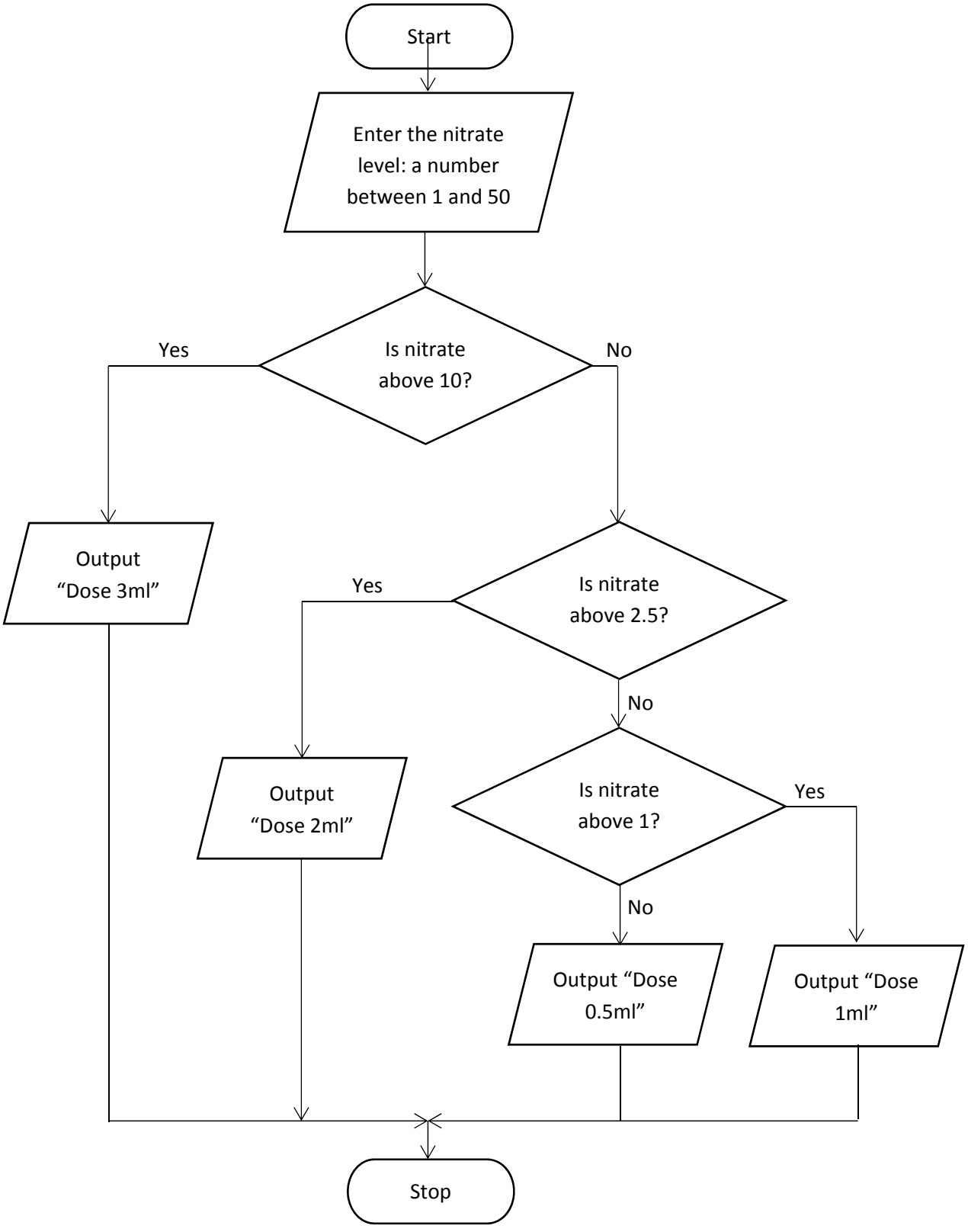
Write a program to display the larger of two numbers entered:



Nitrate Challenge

Difficulty: 🛠🛠 G

When keeping fish, one of the goals to reduce algae is to keep nitrates to a minimum. One way of doing this is to dose a carbon source which nitrifying bacteria within an aquarium consume together with nitrates. The carbon source has to be dosed very precisely. Write this program to determine the dose:



Portfolio grade challenge

Difficulty: 🛠🛠 G

Write a program that inputs a mark from the keyboard for sections of a project: 'analysis', 'design', 'implementation' and 'evaluation'. The program should output the total mark, the grade, and how many more marks were needed to get into the next mark band.

Grades are:

0	U
4	G
13	F
22	E
31	D
41	C
54	B
67	A
80	A*

Cash machine challenge

Difficulty: 🛠🛠 A

A cash machine dispenses £10 and £20 notes to a maximum of £250. Write a program that shows the user their balance, asks them how much to withdraw, ensures this is a valid amount without going overdrawn and with the notes available and outputs the new balance.

Periodic table challenge

Difficulty: 🛠🛠🛠 A

Write a program that asks the user to enter the symbol or name of an element, or group it belongs to. The program should output the name of the element and its atomic weight. E.g.

The user enters Li. The program outputs:

Element: Lithium
Atomic weight: 6.94
Group: Alkali metals

If the user enters Alkali metals, the program outputs the data for all the elements in the alkali metals group.

You only need to get this working for 6 elements from two different groups.

Train ticket challenge

Difficulty: 🛠🛠🛠 A

A train fare from Cheltenham to London is calculated in the following way:

- £20 per station from start to destination for adults.
- £5 extra per stop between 6am and 9am.
- Half price for children.

Write a program that allows the user to enter how many stations they need to pass through, how many adults, how many children and the time of day (as a number: 24 hour clock). The program should output the correct fare.

Test the program with the following data:

Test	Stations	Adults	Children	Time	Expected
1	2	1	0	10	**£40**
2	4	2	2	11	**£240**
3	2	1	0	8	**£50**
4	4	1	0	8	**£100**
5	3	2	1	7	**£187.50**
6	0	0	0	0	**£0**

Hours worked challenge

Difficulty: 🛠🛠🛠🛠 A

Write a program that asks the user for the number of hours worked this week and their hourly rate of pay. The program is to calculate the gross pay. If the number of hours worked is greater than 40, the extra hours are paid at 1.5 times the rate. The program should display an error message if the number of hours worked is not in the range 0 to 60.