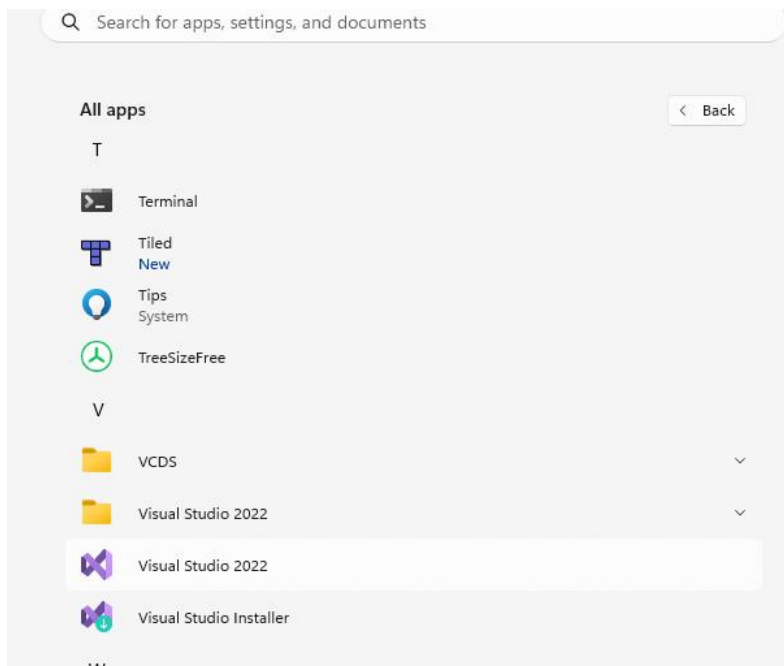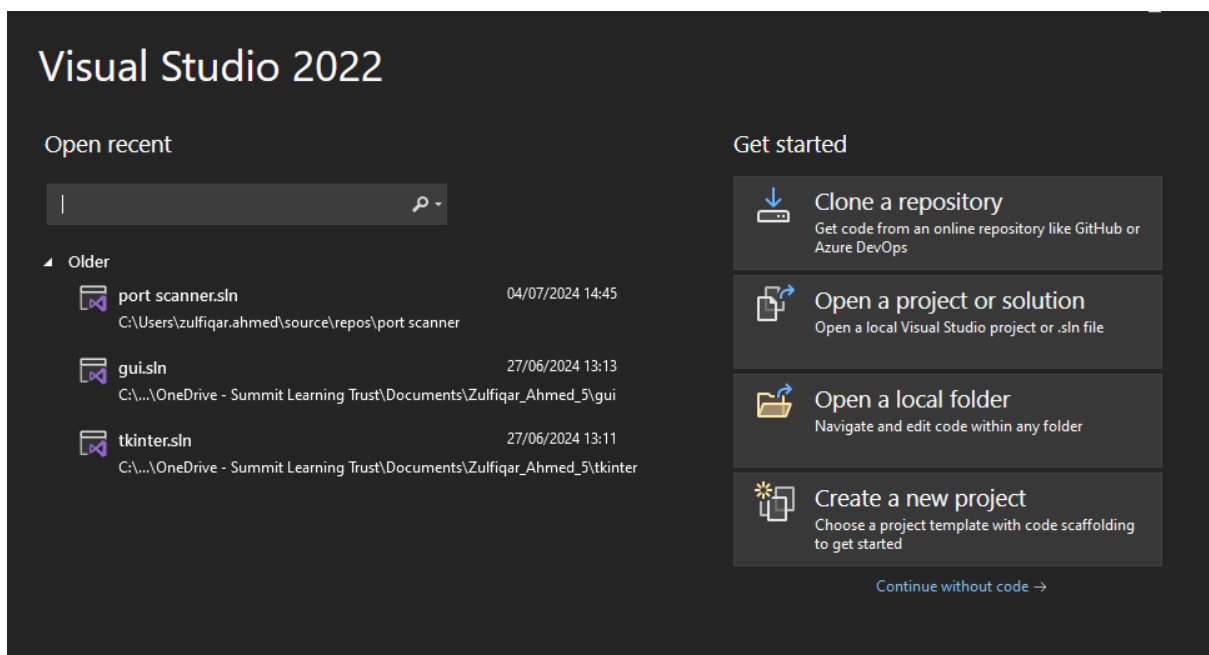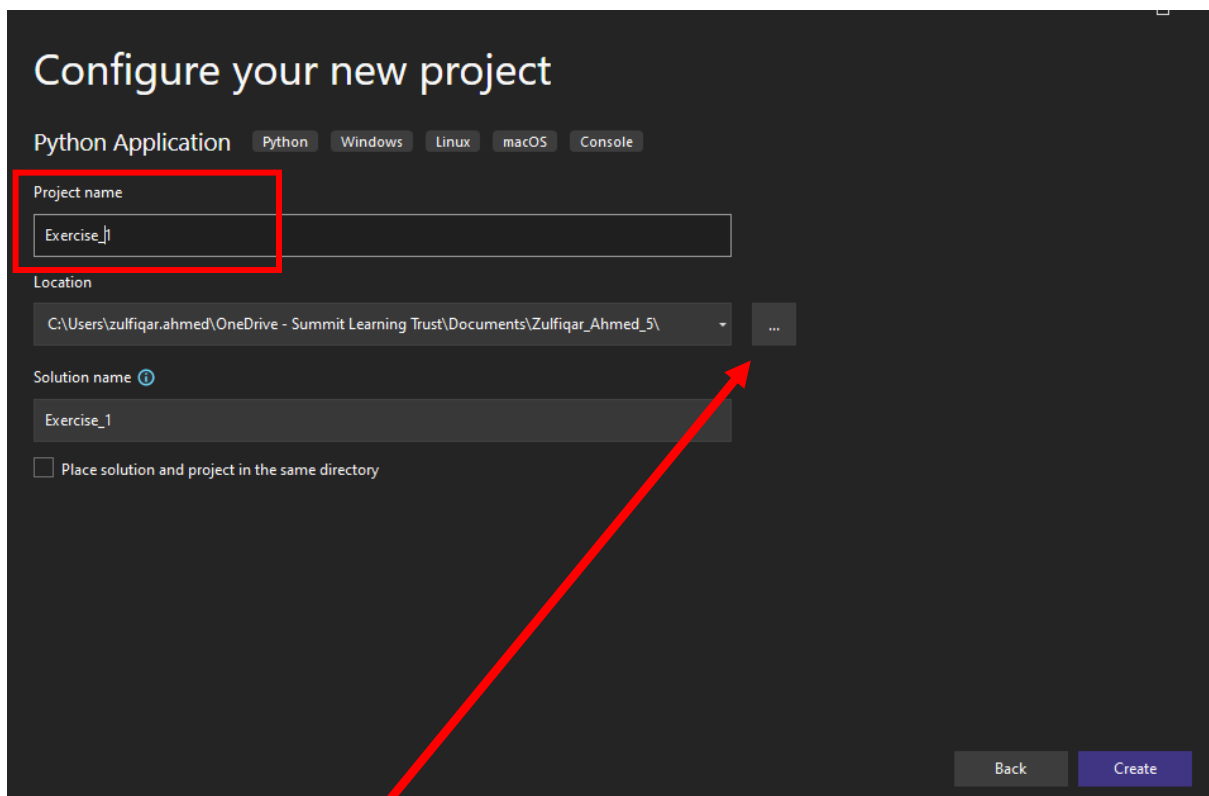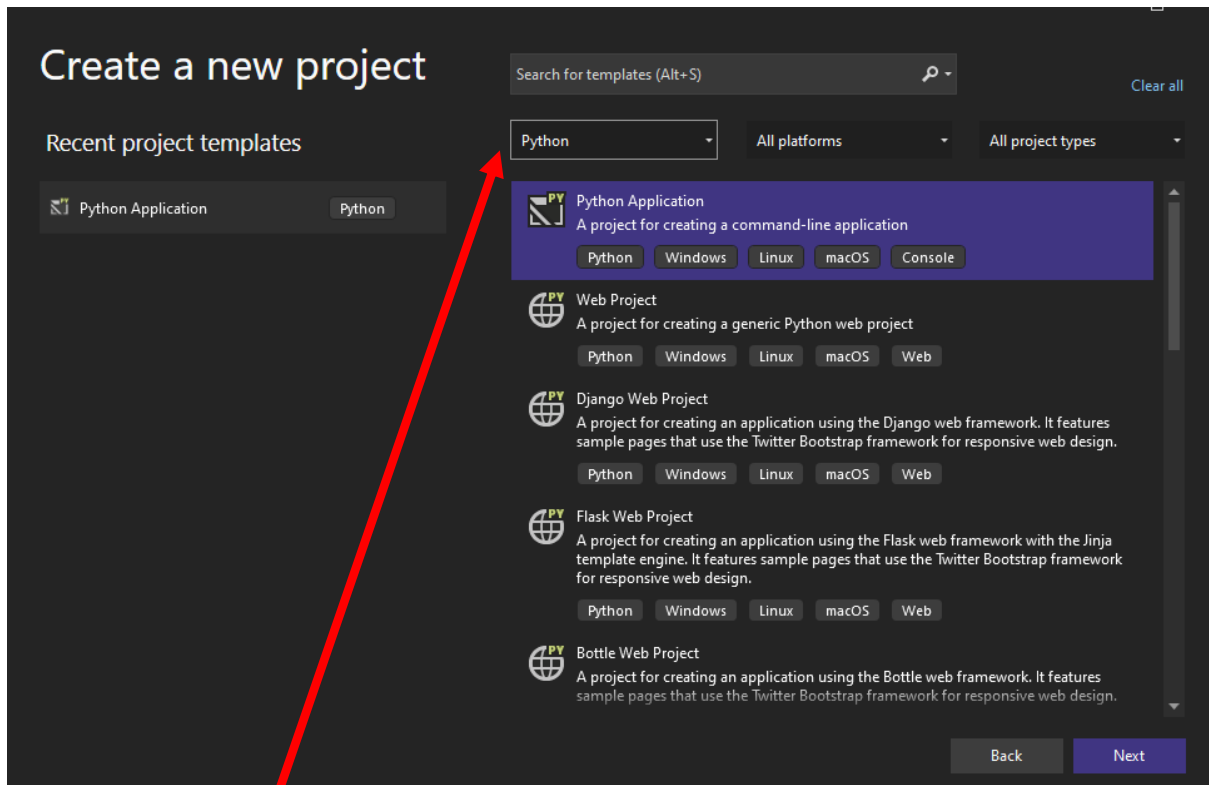# Python Programming Companion Guide

1. Enter visual studio 2022 in the search bar

2. Click on create a new project

Create a new project

Search for templates (Alt+S)

Clear all

Recent project templates

Python Application   Python

Python | All platforms | All project types

Python Application
A project for creating a command-line application
Python   Windows   Linux   macOS   Console

Web Project
A project for creating a generic Python web project
Python   Windows   Linux   macOS   Web

Django Web Project
A project for creating an application using the Django web framework. It features sample pages that use the Twitter Bootstrap framework for responsive web design.
Python   Windows   Linux   macOS   Web

Flask Web Project
A project for creating an application using the Flask web framework with the Jinja template engine. It features sample pages that use the Twitter Bootstrap framework for responsive web design.
Python   Windows   Linux   macOS   Web

Bottle Web Project
A project for creating an application using the Bottle web framework. It features sample pages that use the Twitter Bootstrap framework for responsive web design.

Back   Next

3. In the first drop down box select Python

Configure your new project

Python Application   Python   Windows   Linux   macOS   Console

Project name

Exercise_1

Location

C:\Users\zulfiqar.ahmed\OneDrive - Summit Learning Trust\Documents\Zulfiqar_Ahmed_5\   ...

Solution name ⓘ

Exercise_1

☐ Place solution and project in the same directory

Back   Create

4. Click on the three dots to select where your file will be stored. Then name your file and then click on create.

5. Enter the following code shown above to test application is running. Click on start or press F5 to execute the code.

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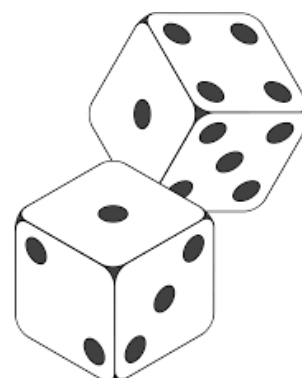
To disable the carriage return, use: `print(x,end='')`

Objective 1: Challenges

Visual dice challenge

Difficulty: ✎ G

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

```
ooooooooooooo
o           o
o   #   #   o
o     #     o
o   #   #   o
o           o
ooooooooooooo
```

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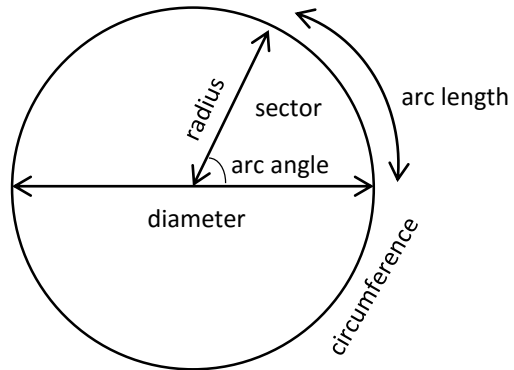
To calculate volume in litres, multiply length by depth by height and divide by 1000.

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).
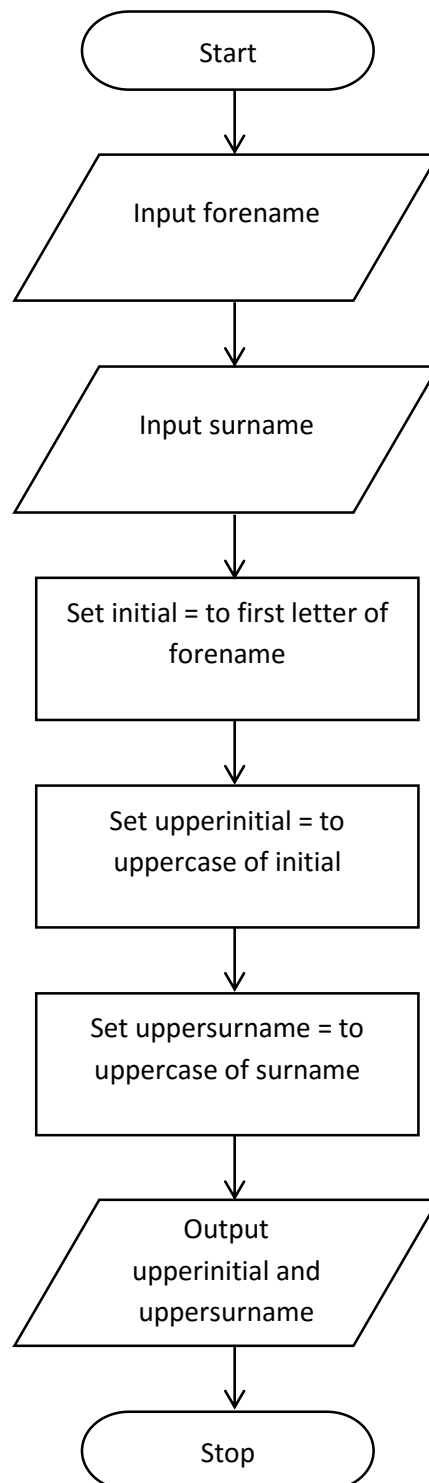
Objective 3: Challenges

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

```
           ┌──────────────┐
           │    Start     │
           └──────────────┘
                  │
                  ▼
         ╱─────────────────╲
        ╱  Input forename   ╲
        ╲                   ╱
         ╲─────────────────╱
                  │
                  ▼
         ╱─────────────────╲
        ╱  Input surname    ╲
        ╲                   ╱
         ╲─────────────────╱
                  │
                  ▼
        ┌────────────────────┐
        │ Set initial = to   │
        │ first letter of    │
        │ forename           │
        └────────────────────┘
                  │
                  ▼
        ┌────────────────────┐
        │ Set upperinitial = │
        │ to uppercase of    │
        │ initial            │
        └────────────────────┘
                  │
                  ▼
        ┌────────────────────┐
        │ Set uppersurname = │
        │ to uppercase of    │
        │ surname            │
        └────────────────────┘
                  │
                  ▼
         ╱─────────────────╲
        ╱     Output        ╲
        ╱  upperinitial and  ╲
        ╲   uppersurname     ╱
         ╲─────────────────╱
                  │
                  ▼
           ┌──────────────┐
           │     Stop     │
           └──────────────┘
```

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

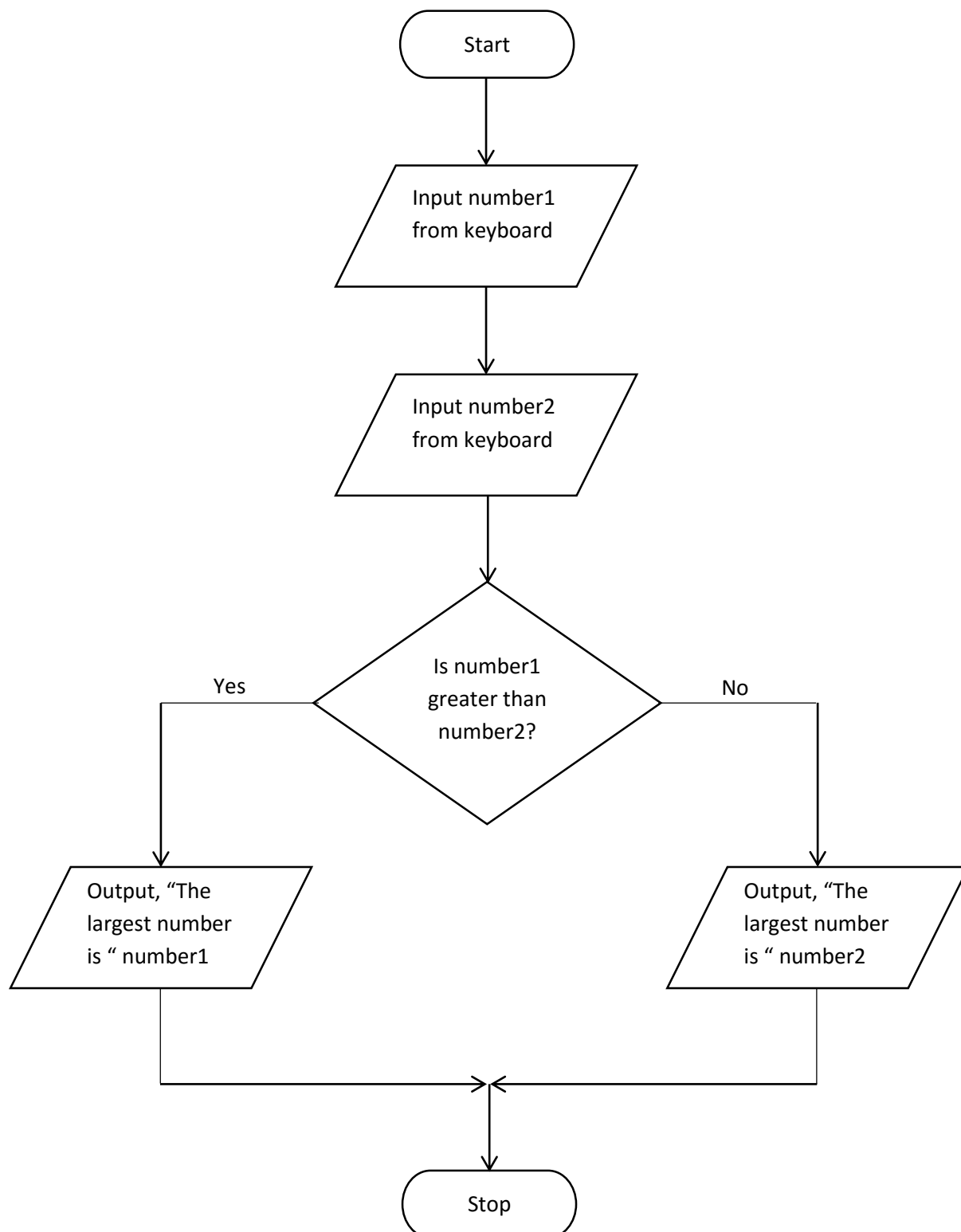Write a program that asks for a number and outputs that number as a graphical dice. E.g.

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

Greatest number challenge

Difficulty: ⚒ G

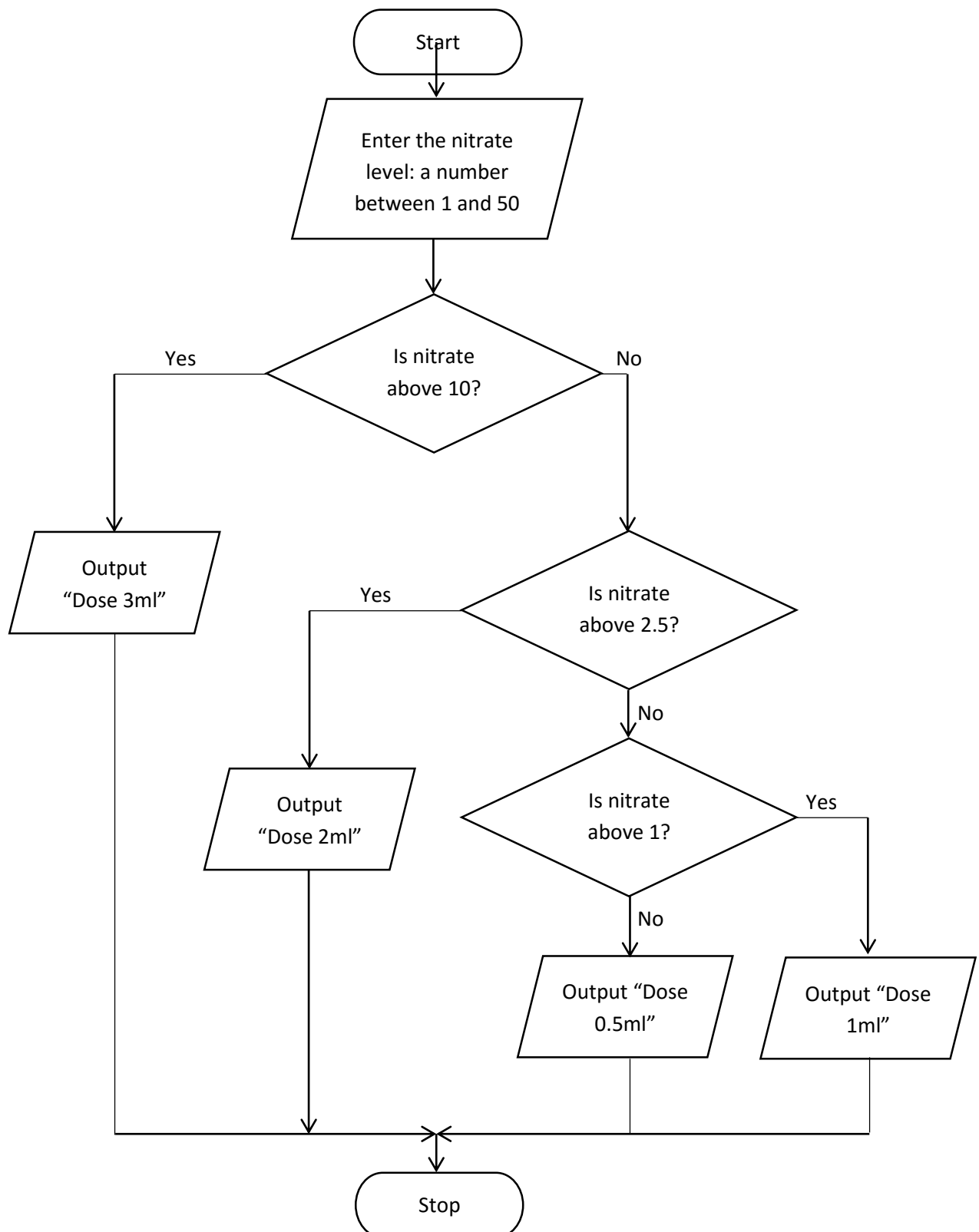Write a program to display the larger of two numbers entered:

```
                    Start

            Input number1
            from keyboard

            Input number2
            from keyboard

                 Is number1
    Yes          greater than          No
                  number2?

  Output, "The                    Output, "The
  largest number                  largest number
  is " number1                    is " number2

                    Stop
```

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:

```
Start
  │
  ▼
Enter the nitrate level: a number between 1 and 50
  │
  ▼
Is nitrate above 10?
  │ Yes → Output "Dose 3ml"
  │ No
  ▼
Is nitrate above 2.5?
  │ Yes → Output "Dose 2ml"
  │ No
  ▼
Is nitrate above 1?
  │ Yes → Output "Dose 1ml"
  │ No → Output "Dose 0.5ml"
  ▼
Stop
```

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.

Objective 5:

Understand how to use arithmetic operations and random numbers

In this objective you learn how to perform mathematical calculations using power, modulus and integer division. Random numbers allow for some unpredictability which is useful for games.

Tasks

1. Try entering the following program to see how arithmetic operators work:

```
#Get user input
number1=int(input("Enter first number: "))
number2=int(input("Enter second number: "))

#Make calculations
power_of_result = number1 ** number2
division_result = number1 / number2
integer_division_result = number1 // number2
modulus_result = number1 % number2

#Output results
print()
print(number1,"to the power of",number2,"is",power_of_result)
print(number1,"divided by",number2,"is",division_result)
print(number1,"divided by",number2,"is",integer_division_result)
print(number1,"divided by",number2,"has a remainder of",modulus_result)
```

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

```
import random
#Roll the dice
random_number = random.randint(1,6)
print("You rolled a",random_number)
```

Note `import random` at the top of the program. For Python to generate a random number it needs the `randint` method which is not in the default set of commands Python understands. Instead it is defined in a library of methods, called 'random'. You can import additional commands from libraries into your program.

3. Change the program so that it outputs a 10 sided dice.

4. Change the program so the user can choose to roll a 4, 6 or 12 sided dice.

Objective 5: Key learning points

How to use arithmetic operations and random numbers

- / and // both perform division. // gives an integer answer.
- **Modulus** is the remainder of a division. E.g. 8 divided by 5 is one with 3 left over (remainder)

Mathematical operators that can be used in conditions include:

+	addition	**	power of
-	subtraction	//	integer division
*	multiplication	%	modulus (remainder after division)
/	division		

Objective 5: Key words

import

Example code: `import random`

Imports a set of library routines into the program to be used.

random.randint

Example code: `x = random.randint(1,y)`

x becomes a random number between 1 and y.

Objective 5: Challenges

RPG dice challenge

Difficulty: ✖ G

In many RPG games, a number of different sided dice are used, not just the standard 6 sided dice. Write a program that asks the user how many sides a dice has and outputs a random number for that dice.

Divisible challenge

Difficulty: ✖✖ G

Write a program that asks the user for two numbers. The program should output whether the two numbers are exactly divisible by each other. If not, it should return the remainder. If the user enters a 0 the program should give an error message. Use the test table below to understand the correct outputs for this program:

Input 1	Input 2	Expected output
3	9	9 is exactly divisible by 3
5	15	15 is exactly divisible by 5
8	23	23 is not exactly divisible by 8, there is a remainder of 7
10	12	12 is not exactly divisible by 10, there is a remainder of 2
0	7	Error: you cannot divide by 0

Month challenge

Difficulty: ✖✖✖ A

Write a program that lets the user enter a number between 1 and 12 and displays the month name for that month number and the number of days in the month. The input 3 would therefore display March has 31 days.
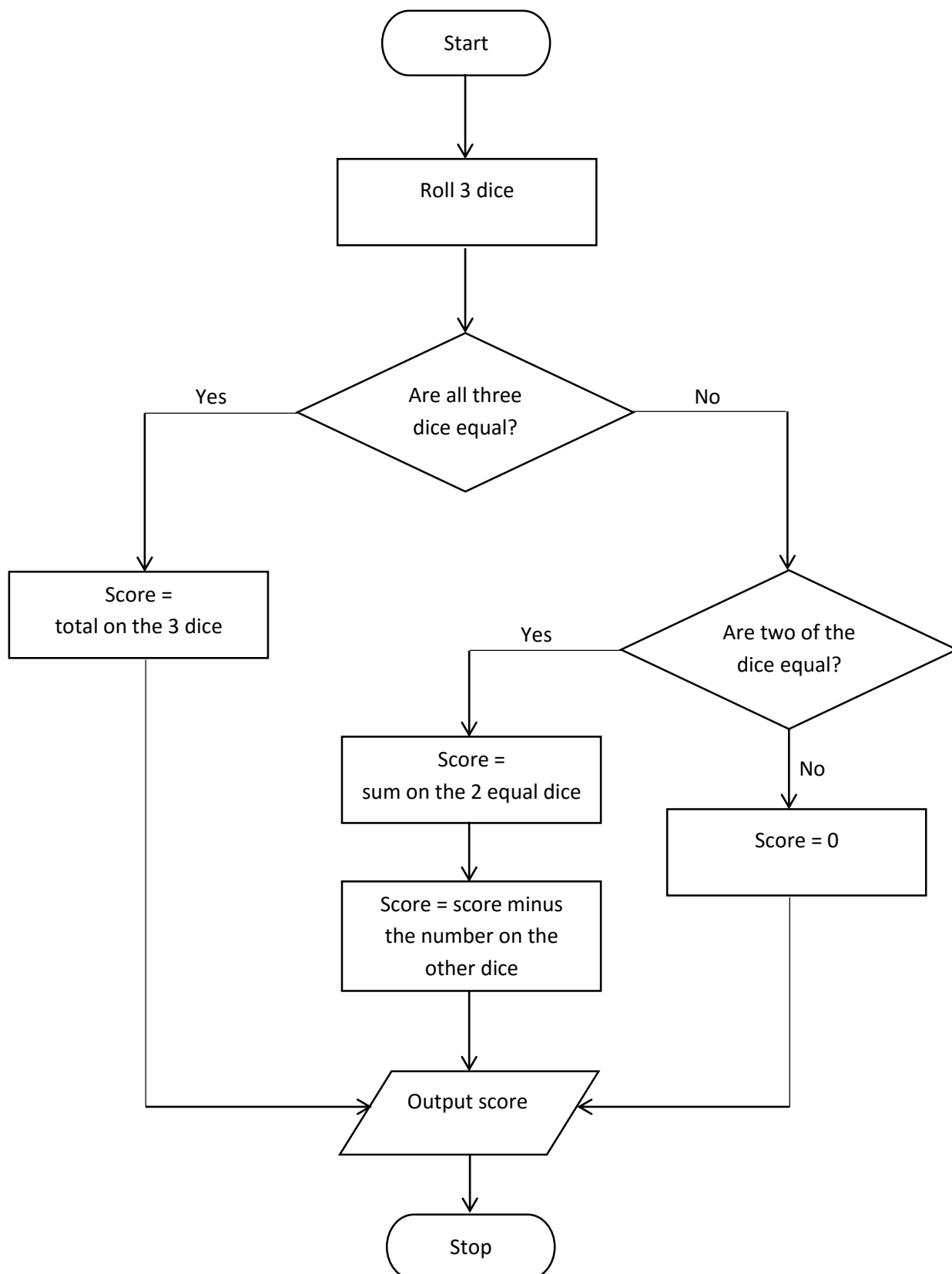
Remember February has 28 or 29 days depending on whether it is a leap year. It is a leap year if the year is exactly divisible by 4.

Dice game challenge

Difficulty: �✗ G

Write a program to calculate the score in this simple dice game:

```
              ┌──────────┐
              │  Start   │
              └──────────┘
                   │
                   ▼
          ┌─────────────────┐
          │   Roll 3 dice   │
          └─────────────────┘
                   │
                   ▼
        ◇ Are all three dice equal? ◇
    Yes ───────────┘        └─────────── No
     │                                    │
     ▼                                    ▼
┌──────────────┐              ◇ Are two of the dice equal? ◇
│   Score =    │          Yes ──┘                    └── No
│ total on the │           │                             │
│   3 dice     │           ▼                             ▼
└──────────────┘    ┌──────────────┐            ┌──────────────┐
     │              │   Score =    │            │  Score = 0   │
     │              │ sum on the 2 │            └──────────────┘
     │              │  equal dice  │                   │
     │              └──────────────┘                   │
     │                     │                           │
     │                     ▼                           │
     │              ┌──────────────┐                   │
     │              │ Score = score│                   │
     │              │ minus the    │                   │
     │              │ number on the│                   │
     │              │ other dice   │                   │
     │              └──────────────┘                   │
     │                     │                           │
     └────────►  ╱ Output score ╲  ◄──────────────────┘
                   │
                   ▼
              ┌──────────┐
              │   Stop   │
              └──────────┘
```

Objective 6:

Understand counter controlled iterations

In this objective you learn how to get code to repeat itself a given number of times without having to enter lots of repeating statements.

Tasks

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

```
for counter in range(5):
    print("This is how you get code to repeat")
```
TAB

2. Try changing the number 5 in the first line as shown below and see what happens.

3. Modify the code as shown below. Run the program and see what happens.

```
for counter in range(7,12):
    print("The counter is",counter)
```
TAB

4. Enter the following commands and see what happens:

```
word="Hello"
for counter in range(0,len(word)):
        print("Letter",counter,"is",word[counter])
```

Note how we can now loop through all the letters in a word by combining an iteration with string manipulation commands. Also note how commands inside an iteration must be indented, but also make the block of code easier to read.

Objective 6: Key learning points
Counter controlled iterations

- An **iteration** is a section of code that is repeated.
- Iterations are also known as **loops**.
- Counter controlled iterations are used when you want to iterate a known number of times.
- Counter controlled iterations use a variable which can be used within the code to know how many times the code has been run.
- Code must be **indented** within an iteration.
- It would be good practice to comment code before an iteration to explain what is happening.
- **Incrementing** a variable increases its value by one.
- **Decrementing** a variable decreases its value by one.
- One 'for' loop can be put inside another 'for' loop. This is known as **nesting**.

Objective 6: Key words

For…

Example code: `For counter in range(0,5):`
`print(counter)`

Executes code within the 'For' structure a known number of times. Counter is the variable that is counting how many times the code has iterated. 0 is the starting value (default). 5 is the stopping value. The starting value does not need to be specified if zero. E.g. `For counter in range(5)`

A third parameter can be used to specify the step. For example to count from 0 to 10 in twos:

```
for counter in range(0,10,2):
    print(counter)
```

By default this is 1 and therefore step 1 is not necessary. To count backwards, a negative value can be used for step. E.g.

```
for counter in range(10,0,-1):
    print(counter)
```

Objective 6: Challenges

Square numbers challenge

Difficulty: ✕ G

Write the program to output all the squares of a number between 1 and 20 as described in the flowchart below.

```
                        ┌──────────┐
                        │  BEGIN   │
                        └──────────┘
                             │
                             ▼
                          ╱──────╲
          ┌─────┐   Yes  ╱ Counter ╲
          │ END │◄───────  start 1  ◄──────────────┐
          └─────┘        ╲ Is = 20? ╱               │
                          ╲──────╱                  │
                             │                      │
                            No                      │
                             ▼                      │
                   ┌──────────────────┐             │
                   │    Squared =     │             │
                   │ counter * counter│             │
                   └──────────────────┘             │
                             │                      │
                             ▼                      │
                   ╱──────────────────╲             │
                  ╱  Output a string    ╲           │
                 ╱  using counter and    ───────────┘
                 ╲  squared variables   ╱
                  ╲ as shown below     ╱
                   ╲──────────────────╱
```
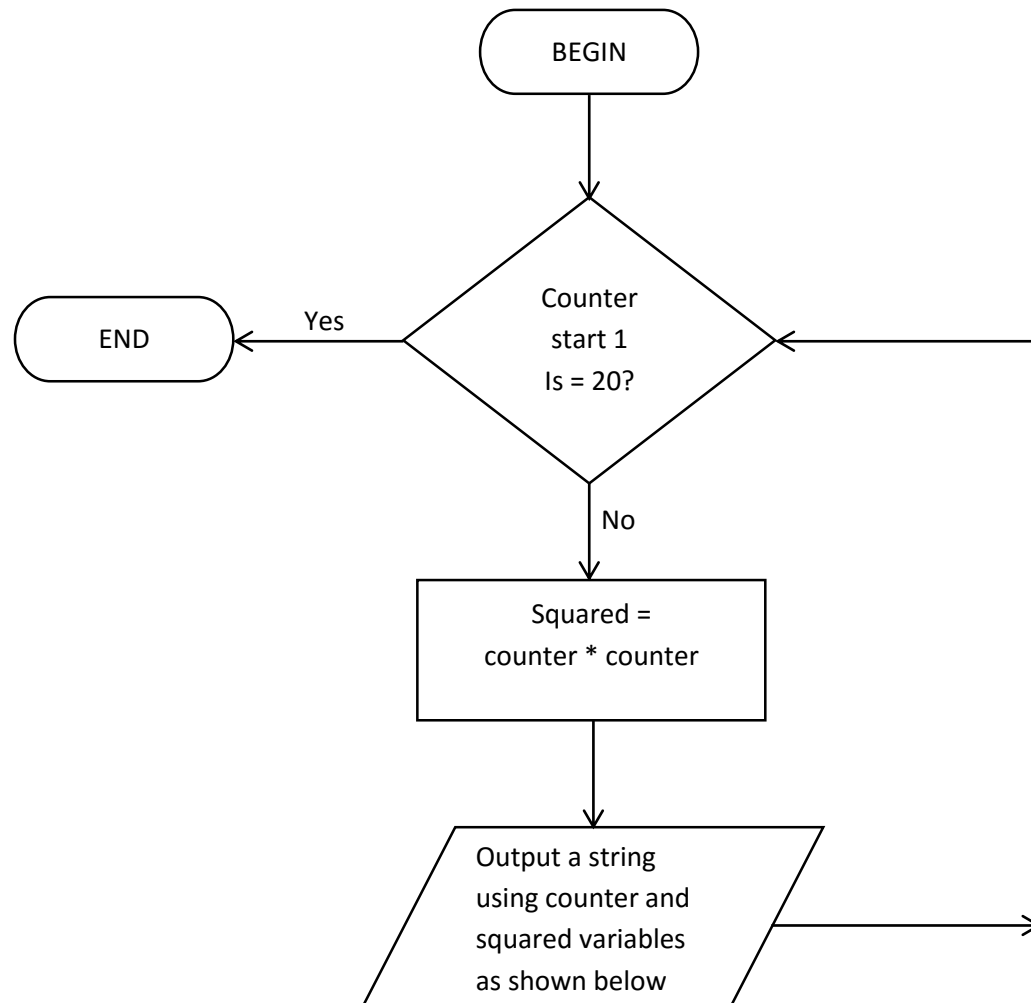
The output should be:

```
    1 squared is 1
    2 squared is 4
    3 squared is 9
    4 squared is 16
```

9 green bottles challenge

Difficulty: ⚒ G

Write a program that prompts the user to enter the number of bottles. The program outputs:

```
9 green bottles sitting on the wall
8 green bottles sitting on the wall
7 green bottles sitting on the wall etc. etc.
```

Times table challenge 1

Difficulty: ⚒ G

Write a program that asks the user to enter a number between 1 and 12. The program outputs the times table of that number between 1 and 12.

Fibonacci sequence challenge

Difficulty: ⚒⚒ G

A number in the Fibonacci sequence is generated by adding the previous two numbers. By starting with 0 and 1, the first 10 numbers will be: 1, 1, 2, 3, 5, 8, 13, 21, 34, 55. Write a program to produce the sequence of 20 numbers.

Average calculator challenge

Difficulty: ⚒⚒ G

Write a program that asks the user to enter how many numbers are to be averaged. The user can then enter the numbers. The program outputs the total and the mean.

FizzBuzz

Difficulty: ⚒⚒⚒ A

Write a program that outputs the numbers from 1 to 100. But for multiples of three output "Fizz" instead of the number and for the multiples of five output "Buzz". For numbers which are multiples of both three and five output "FizzBuzz"

Times table challenge 2

Difficulty: ⚒⚒⚒ A

Write a program that could be used in an infant school to prompt a child with ten simple random 1 digit maths calculation, e.g. 4*7= The user enters their answer and the computer tells them if they are correct. At the end of the test the program outputs how many problems the child got right.

ROT13

Difficulty: ✗✗✗✗ A

ROT13 is a simple letter substitution cipher that replaces a letter with the letter 13 letters after it in the alphabet. ROT13 is a special case of the Caesar cipher, developed in ancient Rome. ROT13 is used in online forums as a means of hiding spoilers, punchlines and puzzle solutions from the casual glance. Create a program that allows the user to enter plain text, and the ROT13 cipher is output. Extend the program to allow cipher text to be input, with the plain text output.

Letter game challenge

Difficulty: ✗✗✗✗ A

A word game awards points for the letters used in a word. The lower the frequency of the letter in the English language, the higher the score for the letter. Write a program that asks the user to input a word. The program should then output the score for the word according to the following rules:

Letter	Points	Letter	Points
E	1	M	14
A	2	H	15
R	3	G	16
I	4	B	17
O	5	F	18
T	6	Y	19
N	7	W	20
S	8	K	21
L	9	V	22
C	10	X	23
U	11	Z	24
D	12	J	25
P	13	Q	26

Objective 7:

Understand condition controlled iterations

In this objective you learn how to get code to repeat continually until a condition is met.

Tasks

1. Try entering the following program and see what happens by entering a few numbers and ending by entering zero:

```
#Program to add up numbers until 0 is entered
#Initialise variables
total=0
number=1
#Enter numbers to add
while number>0:
    number=int(input("Enter a number: "))
    #Add number to total
    total=total+number
#Output result
print("The sum of the numbers is:",total)
```

TAB ⟶ (pointing to `number=int(input("Enter a number: "))`)

Objective 7: Key learning points

Condition controlled iterations

- When it is not known in advance how many times code needs to be repeated, a condition controlled loop is used.
- Condition controlled iterations are useful when you want to validate data entry by only continuing to the next statements once the data input is correct.
- Code must be indented inside iteration statements.

Objective 7: Key words

While…

Example code: `while counter < 3:`

`            …`

Checks the condition and if true executes the indented code, returning to check again afterwards.

Objective 7: Challenges

Menu selection challenge

Difficulty: ⚒ G

Write a program that presents a simple menu to the user: 1. Play Game 2. Instructions 3. Quit
The program should ask the user for a number between 1 and 3 and will output "Let's go" only when a valid entry is made, repeating the menu selection until a valid entry is made.

Compound interest challenge

Difficulty: ⚒ G

Write a program that shows how compound interest grows in a bank savings account. The program should ask the user to input their current balance, the interest rate (0.04 would be 4%) and the desired balance to reach. The program should output the new balance each year until the desired balance is reached. Interest is added to the balance each year.

e.g.

£100 starting balance – 4% interest rate (0.04)
Year 1: New balance = 100 + (100*0.04) = £104
Year 2: New balance = 104 + (104*0.04) = £108.16
Year 3: New balance = 108.16 + (108.16*0.04) = £112.49

Guess the number game challenge

Difficulty: ⚒⚒ G

The computer guesses a number between 1 and 100. The player has to try and guess the number in as few attempts as possible. When the user enters a number they are told if the guess is too high or too low until the number is guessed correctly. The player is told how many guesses they made. Write a program to play the game.

Gamertag challenge

Difficulty: ✘✘ G

A gamertag is a username on a typical games console that other players can identify the player with. There are some limitations to them, such as, it cannot be longer than 15 characters.

Write a program to input a valid gamertag and keep asking for a new gamertag until a valid entry is made.

Pseudocode
1.0 Write a comment to explain the program checks the length of gamerTags entered.
2.0 Set a variable called valid_gamertag = true
3.0 Start a condition controlled iteration to check valid_gamertag
4.0 Output a message on the screen to say gamertags must be less than 15 characters.
5.0 Get a gamertag from the keyboard and put it into a variable called gamertag.
6.0 Using the function len() put the number of characters in gamertag in a variable called gamertag_length.
7.0 Check if the length of gamertag is more than 15 characters by comparing gamertag_length to the number 15.
7.1 Output a message to say 'your gamertag is too long' if it is of an invalid length.
7.2 Output a message to say 'gamertag accepted' if it is of a valid length.
8.0 End iteration

Rock, paper, scissors challenge

Difficulty: ✘✘✘ A

The computer and player choose one of rock, paper, or scissors. The output of the encounter is then displayed with rock beating scissors, scissors beating paper, and paper beating rock. The winner scores 1 point for a win. The score for both players should be output. The game is won when one player reaches 10 points.

Happy numbers challenge

Difficulty: ✘✘✘ A

Write a program to output whether a chosen number is happy or sad. A happy number is a number defined by the following process: Starting with any positive integer, replace the number by the sum of the squares of its digits, and repeat the process until the number either equals 1, or it loops endlessly in a cycle which does not include 1. Those numbers for which this process ends in 1 are happy numbers, while those that do not end in 1 are unhappy numbers (or sad numbers). When the algorithm ends in a cycle of repeating numbers, this cycle always includes the number 4.

For example, 19 is happy, as the associated sequence is:

$1^2 + 9^2 = 82$
$8^2 + 2^2 = 68$
$6^2 + 8^2 = 100$
$1^2 + 0^2 + 0^2 = 1.$

XP Challenge

Difficulty: ✗✗✗ A

In many online multiplayer games, players are awarded experience points (known as XP) for completing a variety of objectives in the game. When the player has accumulated enough XP, their rank increases (they rank up) and their XP resets to zero (although extra XP is carried over). Each rank requires more XP than the one before.

Write a program that:

- Allows the user to keep entering XP for a game until it is 2000 or more.
- When the XP reaches 100, 'Promoted to Corporal' is output.

Pseudocode
1.0 Set the player's total XP to be 0
2.0 Set the player's rank to be 1
3.0 Loop while the player's XP is less than 2000
3.1 Ask the user to enter the XP earned in the last game
3.2 Add the XP earned from the last game to the total XP
3.3 Check if the XP earned is more than that required to be a Corporal and that their rank is 1
3.3.1 Tell the player they have been promoted
3.3.2 Set XP back to zero but carry over any XP left
3.3.3 Set player's rank to be 2
3.4 Display the total XP

- Extend the program so that the player ranks up according to the table below:

XP needed	Rank awarded
0	Private
100	Corporal
300	Sergeant
700	Staff Sergeant
1500	Warrant Officer

Objective 8:

Understand subroutines, procedures and functions

In this objective you learn how to divide your program into more manageable sections to make it easier to read and reduce the need for entering repeating code.

Tasks

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

```python
#Global variables accessed by all subroutines
MaxNoOfStars=0
NoOfStars=0
NoOfSpaces=0

#Set the size of the pyramid of stars
def InitialiseNoOfSpacesAndStars():
    global NoOfSpaces, MaxNoOfStars, NoOfStars
    MaxNoOfStars=int(input("How many stars at the base (1-40)? "))
    #Calculate spaces needed from tip
    NoOfSpaces = MaxNoOfStars // 2
    #Set tip of pyramid to one star
    NoOfStars = 1

#Outputs spaces before stars
def OutputLeadingSpaces():
    global NoOfSpaces
    for count in range(NoOfSpaces):
            print(" ",end="")

#Outputs stars
def OutputLineOfStars():
    global NoOfStars
    for count in range(NoOfStars):
        print("*",end="")
    #Move to next line
    print()

#Adjusts number of stars & spaces after output
def AdjustNoOfSpacesAndStars():
    global NoOfSpaces, NoOfStars
    NoOfSpaces = NoOfSpaces - 1
    NoOfStars = NoOfStars + 2

#Main program starts here
InitialiseNoOfSpacesAndStars()
while NoOfStars <= MaxNoOfStars:
    OutputLeadingSpaces()
    OutputLineOfStars()
    AdjustNoOfSpacesAndStars()
```

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

```python
#Program to output a set of random numbers
import random
#Output random numbers
def outputrandoms(n,m):
    for counter in range(1,n+1):
        randomnum = random.randint(1,m)
        print("Number",counter,"is",randomnum)

#Main program starts here
number=int(input("How many numbers do you want to output? "))
maximum=int(input("What is the maximum number? "))
outputrandoms(number, maximum)
```

There are two ways to share data between subroutines. In the first program, the variables were initialised (given a starting value) outside of any subroutine. That means they are global variables to the program, and their values can be accessed, shared and changed by any subroutine in the program, using the command 'global'.

In the second program the variables 'number' and 'max' are passed into the subroutine outputrandoms. The values of the variables become n and m in the subroutine outputrandoms. The variables n and m are different variables to number and max, but the value stored in number and max was passed into n and m. . In this example, the variables n and m are lost at the end of the outputrandoms subroutine. That means they are local variables to the subroutine.

3. Try entering this program and see what happens:

```python
#How functions can be used
#Returns a positive number from subtraction
def floor(a, b):
    f = a - b
    if f > 0:
        return f
    else:
        return 0

#Main program starts here
num1=int(input("Enter the first number: "))
num2=int(input("Enter the second number: "))
print("The floor number is:",floor(num1, num2))
```

4. Change the function so it receives the length of two sides of a right angled triangle, and returns the length of the hypotenuse (h2 = o2+a2)

Objective 8: Key learning points
Understand subroutines, procedures and functions

- Subroutines are **sections of code** to break a longer programs into smaller pieces. Thereby making them easier to read and more manageable for teams of programmers to work together on one program.
- Subroutines are also known as **procedures**.
- **Functions** carry out small tasks on data by taking one or more **parameters** and **returning** a result.
- Subroutines/procedures may also take parameters, but do not return a result.
- It is good practice to comment each subroutine or function to explain its purpose.
- Variables declared outside of all subroutines are known as **global variables**. They are available to all subroutines and functions.
- Global variables are not memory efficient since they hold memory for the entire time a program is running. They should therefore be kept to a minimum.
- Variables declared inside a subroutine or function are known as **local variables**. They lose their value when the routine ends, releasing memory.
- Passing variables between subroutines is known as **parameter passing**.

Objective 8: Key words
def…
Example code: `def maximum (x,y):`
```
                …
```

Creates a subroutine called max. Two parameters are passed into max, with local variables x and y holding their values.

max would be called with: x=`maximum(6,9)`
6 would be passed as a parameter to variable x. 9 would be passed as a parameter to variable y.

global
Example code: `def maximum (x,y):`
```
                global z
                …
```

Includes a global variable in the subroutine.

return
Example code: `def maximum (x,y):`
```
                if x>y:
                        return x
                else:
                        return y
```

Returns a value to the calling routine. E.g. x = `maximum(4,7)` would result in x having the value 7.

Objective 8: Challenges

VAT challenge

Difficulty: � G

Write a program that asks the user for a price of an item. Include a function that returns the VAT for the item. This should be output in the main program.

Conversion challenge

Difficulty: ✂✂ G

Write a program that provides a conversion utility, converting from metric to imperial measurements. The list of possible conversions can be presented to the user in a menu.

Each conversion should be a separate subroutine or function to allow the program to be modular and added to later.

Conversion challenge part 2

Difficulty: ✂✂✂✂ A

Improve the conversion program to use a 'text parser'. Type 25 cm in inches into Google and see what happens. The program should be able to evaluate a string input and still work.

Darts challenge

Difficulty: ✂✂ G

Darts is a game where the objective is to reach a score of zero starting from 501. A player throws 3 darts and their total from the throw is subtracted from their score. A player has to achieve a score of exactly zero to win the game. A throw that would result in a score below two does not count.

Create a program to allow the user to enter a total from their darts. The program should then subtract this from their current score if the remaining score would be greater than one.
The player should win the game when their score becomes exactly zero.

Pseudocode
1.0 Set the player's score to be 501
2.0 Display the message: New game. Player starts at 501 points.
3.0 Loop while the player's score does not equal zero
3.1 Ask the user to enter the total from 3 darts
3.2 Check if the player's score minus the total is greater than 1
3.2.1 Subtract the total from the player's score
3.2.2 Display the player's new score
3.3 Else check if the player's score minus the total equals 0
3.3.1 Subtract the total from the player's score
3.3.2 Display the message: Player 1 wins the game

Darts challenge part 2

Difficulty: �✕✕ A

Allow the player to start a new game when they have won.

Create a two player version of the game where players take it in turns to throw darts and have the total subtracted from their own current score. The player that reaches zero first wins.

Snake eyes challenge

Difficulty: ✕✕✕

Write a program that plays the following game:

- Player 1 starts the game by rolling two dice.
- The total is the value of the two dice. E.g. 3 and 4, total = 7
- If a player rolls a single 1 on either die, they lose the running total, it is reset to 0 and play passes to the other player without them having a choice to gamble or bank.
- If a player rolls a double 1, they lose their banked total and running total, both are reset to 0 and play passes to the other player without them having a choice to gamble or bank.
- If a 1 was not thrown on either die, the player has the option to gamble and roll the dice again or to bank the accumulated score.
- If the player gambles, the total from the two dice is added to a running total and the dice are rolled again, creating a new total.
- If a player banks, the running total is added to the banked score and the running total is reset back to 0.
- If a player banks, his turn is over and play passes to the other player.
- The game is won by the first person reaching a banked total of 100 or more.

Pass the Pigs challenge

Difficulty: �ख✖✖✖ A

Pass the pigs is a game where a player tosses two plastic pigs and scores points according to how they land. If the two pigs land on the same side, that is known as a 'sider' and the player scores 1 point. If one pig lands on its feet, this is known as a 'trotter' and the player scores 5 points. Both pigs landing on their feet is a 'double trotter' and scores 20 points. After each throw the player can throw again to increase their score. However, if both pigs land on opposite sides that is a 'pig out' and the player's score is reset to 0. The player attempts to score 100 points or more in as few throws as possible.

Pseudocode
1.0 Set player's score to be 0
2.0 Loop while the player's score is less than 100
2.1 Wait for the user to press enter
2.2 Output a blank line
2.3 PigA is a random number between 1 and 3
2.4 PigB is a random number between 1 and 3
2.5 Check if PigA and PigB land on the same side
2.5.1 Output 'sider' – 1 point
2.5.2 Increase player's score by 1
2.6 Check if PigA and PigB both land on the other side
2.6.1 Output 'sider' – 1 point
2.6.2 Increase player's score by 1
2.7 Check if PigA and PigB land on the opposite sides
2.7.1 Output 'pig out' – back to 0 points
2.7.2 Set player's score to 0
2.8 Check if PigA and PigB land on the opposite sides other way
2.8.1 Output 'pig out' – back to 0 points
2.8.2 Set player's score to 0
2.9 Check if PigA but not PigB lands on its feet
2.9.1 Output 'trotter' – 5 points
2.9.2 Increase player's score by 5
2.10 Check if PigB but not PigA lands on its feet
2.10.1 Output 'trotter' – 5 points
2.10.2 Increase player's score by 5
2.11 Check if PigA and PigB both land on their feet
2.11.1 Output 'double trotter' – 20 points
2.11.2 Increase player's score by 20
2.12 Output player's score

Pass the Pigs challenge part 2

Difficulty: ✖✖✖✖ A

Include these additional point scores:

Snouter	Pig lands on its nose	5 points
Double Snouter	Both pigs land on their nose	20 points
Razorback	Pig lands on its back	5 points
Double Razorback	Both pigs land on their back	20 points

The game is now unbalanced because it is easier to score points than to pig out or score a sider. Change the balance of the game so it remains a challenge. Generate a higher random number and use a greater or lesser range of numbers to represent each position.

Pass the Pigs challenge part 3

Difficulty: ✖✖✖✖ A

Create a two player version of the game where players can choose to bank after each throw, giving their opponent a chance to throw. Players can continue their go until they pig out or choose to bank. The first to 100 banked points wins.

Research the full range of scores available in the real game 'pass the pigs'. Include these in your game with an appropriate balance of probability.

Objective 9:

Understand arrays and lists

In this objective you learn how to store multiple items without having lots of different variables.

Tasks

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

```
#Declare list and data set
sentence = ["The","quick","grey","fox","jumps"]
#Output contents of list
print(sentence[0])
print(sentence[1])
print(sentence[2])
print(sentence[3])
print(sentence[4])
```

Sentence is an example of a list, or one dimension array: a group of elements with the same name and data type.

List name:	sentence				
Index:	0	1	2	3	4
Data:	The	quick	grey	fox	jumps

2. Try adding this code, enter the colour brown and see what happens:

```
sentence[2] = input("Enter new colour: ")
print(sentence)
```

Note how it is possible to change an individual element of an array by referring to the index.

3. You don't always want to populate the array with data when it is declared. For example, to declare an array of 100 elements, you would use the alternative syntax:

```
item = []
for counter in range(100):
        item.append(0)
```

You can then put data into the array using the syntax:

```
item[0] = "Alpha"
item[99] = "Omega"
```

Note that indexes always start at zero. So, one hundred elements is 0-99.

4. The power of lists often comes by combining them with iterations.
   Enter this code as an alternative to the first program and notice how the amount of code is much reduced when you use a for loop to output each of the indexes of the list.

```python
sentence = ["The","quick","brown","fox","jumps"]
for index in range(0,4):
```
TAB ⟶
```python
    print(sentence[index])
```

5. Try entering this program and test by entering 8743 when prompted for a product code:

```python
#Declare array and set data
product = ["1262", "Cornflakes", "£1.40", "8743", "Weetabix", →
"£1.20", "9512", "Rice Krispies", "£1.32"]
product_code=0  #Product code to find
found=False      #Whether product was found in array

#Ask use for the product code to find
product_code=input("Enter the product to find: ")

#Use a loop to cycle through the indexes
counter = 0
found = False
while found == False and counter != 9:
    #Output the product if found
    if product[counter] == product_code:
        print(product[counter + 1])
        print(product[counter + 2])
        found = True
    counter = counter + 1

#Output message if product is not found
if counter == 9:
    print("Product not found")
```

Note how an iteration and Boolean variable can be used to go through all the indexes of an array to find an item.

This is not the best way to write this program, but does illustrate how a number of programming concepts can be combined.

You can do a lot more with lists in Python when you combine them with methods.

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

```
#Create a new list
inventory = []
#Add items to the list
inventory.append("torch")
inventory.append("gold coin")
inventory.append("key")

#Remove items from the list
inventory.remove("gold coin")

#Sort the items into alphabetical order
inventory.sort()

#Output all the items in the list
for counter in range (len(inventory)):
    print(inventory[counter])
```

7. Change the program so that a 'pick axe' is added to the inventory.

8. Change the program so the inventory is output and the user can choose which item to drop.

Objective 9: Key learning points
Understand arrays and lists

- Lists are **dynamic data structures** because the size of the list changes when the program is running.
- A list has an **identifier**: the name of the array.
- A list has an **index** which points to one element of the list.
- Indexes start at 0.
- A list has multiple **elements**: the data stored in it.
- Lists have **methods** that are operations that can be performed on the list with no additional code written by the programmer.

Objective 9: Key words
=[]

Example code: `x = []`

Creates a single dimension list called x.

Data can be populated into a list using square brackets and commas to separate items:

`x = [4, 7, 3, 1, 9]`

Empty lists must be populated before items can be added at specific indexes. For example, to initialise a list of 100 indexes:

```
item = []
for counter in range(100):
        item.append(0)
```

list.append

Example code: `listname.append(item)`

Adds item to the list.

list.insert

Example code: `listname.insert(index,item)`

Inserts item at position index.

list.remove

Example code: `listname.Remove(item)`

Removes item from the list.

list.pop(x)

Example code: `listname.pop(index)`

Removes an item from the list, stored at the index. Indexes start at 0 with the first item.

list.sort

Example code: `listname.sort()`

Sorts all the items in a list.

list.extend

Example code: `listname.extend(y)`

Appends list y to a list.

list.reverse

Example code: `listname.reverse`

Reverses the order of items in a list.

del

Example code: `del listname[:]`

Removes all the items from a list.

in

Example code: `if x in listname:`

Returns whether the list contains item x.

Objective 9: Challenges

Text dice challenge

Difficulty: �֍ G

Write a program that generates a random number and outputs the word, "one", "two", "three", "four", "five" or "six" without using a condition statement like 'if'.

Notebook challenge

Difficulty: ✖✖ G

Write a program that allows the user to store up to 10 notes. The program should have this functionality on a continual loop:

1. Output all the notes stored, numbered 0-9.
2. Ask the user to enter a number for a note to change.
3. Ask the user to enter the new note.
4. Over-write any existing note stored at that position.

Currency converter challenge

Difficulty: ✖✖ G
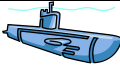
Write a program that stores the names and exchange rates of currencies in an array. The program should ask the user to enter the number of British pounds and a currency to convert into. The program should then output the exchange rate followed by the conversion from British pounds.

Battleships challenge

Difficulty: ✖✖ G

Battleships is a game for two players. Each player places a number of ships on a board but does not reveal their location to their opponent. Each ship occupies one or more adjacent squares. Each player takes it in turn to say aloud a number on the grid. The player scores a hit if the number matches a space occupied by a ship, or a miss if it does not. The player to sink all their opponents ships first wins.

Part of the 1D Battleships game board:

The computer randomly chooses locations between 1 and 50 for five ships. Each ship only takes up one square (index). The player attempts to sink the computer's ships in as few moves as possible.

Create a one player game of battleships. The player does not have ships, and the computer does not fire at the player.

Battleships challenge part 2

Difficulty: ✖✖✖ A

1. Keep score to tell the player how many hits and misses they have had.
2. The player should not be able to choose the same index twice.

Battleships challenge part 3

Difficulty: ✖✖✖✖ A

The game is two player. The player positions their ships for the computer to sink. The computer chooses a random number to fire at.
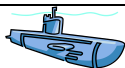
Battleships challenge part 4

Difficulty: ✖✖✖✖ A

1. Ships should occupy 3 indexes, not just one, within the boundaries of the board.
2. One ship occupies 5 indexes, one is 4 indexes, one is 3 indexes and one is 2 indexes.

Battleships challenge part 5

Difficulty: ✖✖✖✖ A

1. Change the game so it is played on a 10x10 grid like this:

	A	B	C
1			
2			
3			

Battleships challenge part 6

Difficulty: ✖✖✖✖ A

The computer chooses indexes to fire at more methodically. If one index is a hit then it should fire at the one before or the one after next until the ship is sunk.

Objective 10:

Understand serial files

In this objective you learn how to read and write data to files so it can be saved and loaded again later.

Tasks

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

```
#Open a text file for writing
text_file = open("data.txt", "w")
text_file.write("This is a simple way to save data\n")
text_file.write("one line at a time")
text_file.close()
print("File created.")
```

You should find a new file called data.txt in the same folder that you saved the program in. Open it and examine the contents.

Note the **"w"** in line 2. This means write data to the file.

Note the \n in line 3. This is known as an escape sequence and adds a carriage return to start a new line.

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

```
#Open a text file for reading
text_file = open("data.txt", "r")
print(text_file.read())
text_file.close
```

3. Change the program so that 3 lines of text are written to a file and read back in again.

It is often useful to read in all the data from a file line by line until you reach the end of the file. To do this we would want to combine an iteration with a file read.

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

```python
#Open a text file for reading
text_file = open("data.txt", "r")

#Read each line into a variable and output
while True:
    a = text_file.readline()
    if not a: break
    print(a)
```

Note how we can use an infinite loop and a break command to read all the lines in the file until no more lines are found.

Each line in the text file has a hidden \n new line escape sequence to indicate the next item of data is on a new line. This has to be removed before selection statements can be used because C is not the same as C\n.

The method .strip will remove the escape sequence from the string. E.g.

```python
x = text_file.readline()
x = answer.strip()
```

Objective 10: Key learning points

Understand serial files

- Serial files store data with no order to the data maintained.
- To search data from a serial file you begin at the start of the file and read all the data until the item is found.
- Data cannot be deleted from a serial file without creating a new file and copying all the data except the item you want to delete.
- Data cannot be changed in a serial file without creating a new file, copying all the data across to a new file, inserting the change at the appropriate point.
- Data can be appended to a serial file.
- A file can be open for reading or writing, but not reading and writing at the same time.
- Serial files are quite limiting, but are useful for simple data sets and configuration files. Other types of files include: sequential files where order of the data is maintained, index sequential files for large data sets and random files which allow you to access any item without searching through the file from the start.

Objective 10: Key words

open()

Example code: `text_file = open("filename", "w")`

Opens a file specified by the path and filename.

`"r"`   - for reading data
`"w"`   - for writing data, over-writing a file that may already exist
`"a"`   - for adding data to the end of the file without overwriting existing contents

textfile.read()

Example code: `x = textfile.read()`

the contents of the file is copied to x.

textfile.readline()

Example code: `variable = LineInput(1)`

Reads a single line of data from an open file into x.

textfile.write()

Example code: `textfile.write("data to write")`

Writes a single item of data to an open file.

textfile.close

Example code: `textfile.close`

Commits any data in the file buffer to storage and closes the file. If a file is not closed, data will not be written.

Objective 10: Challenges

Quote of the day challenge

Difficulty: ⚒ G

Write a program that outputs a 'quote of the day', stored in a text file, created in Notepad.
"If you fell down yesterday, stand up today"

Quote of the day challenge 2

Difficulty: ⚒⚒ G

Implement this enhancement:

1. The program outputs a random quote of the day from a file of quotes.

Product catalogue challenge

Difficulty: ⚒ G

A product has a code, description and price. Write a program that asks the user for the data for a product, and saves the data in a serial file.

Product catalogue challenge 2

Difficulty: ⚒⚒ G

Modify the program to continue to ask for products until the user does not enter a product code. All products should be saved in the file.

Product catalogue challenge 3

Difficulty: ⚒⚒⚒ A

Create a simple menu that allows the user to enter a new catalogue, one new product to add to the end of the catalogue, or to output the items in a catalogue.

Quiz challenge

Difficulty: ⚒⚒⚒ A

Q10 is a simple quiz game. Ten questions are presented to the player one at a time, together with 4 possible answers. The player has to choose which answer: A, B, C or D is correct. The computer outputs the correct answer after the player has answered.

Create the following sample quiz file containing two questions in Notepad and save it in a folder for your new quiz program. Make sure you name it questions.txt

questions - Notepad

```
Which birthstone is associated with the month of May?
A: Diamond
B: Ruby
C: Emerald
D: Sapphire
C
Which two colours are on the flag of Poland?
A: Red and Green
B: Blue and White
C: Green and White
D: Red and White
D
```

Note the file contains the question, the four possible answers followed by the correct answer.

Create the quiz program to use this file.

Pseudocode
Open the questions and answers text file for reading
Read in the question
Output the question to the screen
Read in 4 answers one by one
Output each answer to the screen
Read in from the file which is the correct answer
Ask the user to enter their answer: A, B, C, D
Output the correct answer
Close the text file

Quiz challenge 2

Difficulty: �ख✖✖ A

Implement these enhancements:

1. The program outputs if the user got the question right or wrong.
2. The program keeps score and outputs the score to the user.
3. Modify the program to work for ten questions. Use a loop to show the questions one at a time, you don't need to duplicate the code!
4. The program shows the question number 1-10.

Quiz challenge 3

Difficulty: ✖✖✖✖ A

Implement these enhancements:

1. The player is only allowed to answer the next question if they got the previous one correct. When an incorrect answer is given the quiz ends.
2. The program picks a random set of questions stored in different files.
3. The program prevents an illegal data entry. Only characters A-D are accepted.

Till challenge

Difficulty: ✖✖✖✖ A

Using the product catalogue program, create a program to meet the following specification:

1. A till can be put into 'admin' or 'trading mode'.
2. In the trading mode, the user can perform the following operations:
    a. Enter a product code. The product is found and the description and price is displayed.
    b. The price of the product is added to a sub-total.
    c. When a product code is not entered, the total is shown.
    d. The user enters how much money was given and the change is shown.
    e. A new set of transactions then begins.
3. In the admin mode, the user can perform the following operations:
    a. View the product catalogue
    b. Add a new product
    c. Edit a product
    d. Delete a product

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.

More Challenges

Change calculator

Difficulty: ✖✖✖ A

Write a program that outputs the change to be given by a vending machine using the lowest number of coins. E.g. £3.67 can be dispensed as 1x£2 + 1x£1 + 1x50p + 1x10p + 1x5p + 1x2p.

Parser challenge

Difficulty: ✖✖✖✖ A

Write a program that allows the user to enter a calculation in the format: 16+8. There may be up to 5 digits for each number. The operator may be a +, - * or /. The program should perform the calculation and output the answer. The user should not have to choose which operator they are going to use. They should enter the calculation as one string.

Blackjack challenge

Difficulty: ✖✖✖✖ A

Create a program that plays a game of Blackjack between the player and computer. The program should:

- Deal two cards to the player and computer. One of the dealers cards is shown face up. All other cards are dealt face down.
- Cards 2-10 are face value. All picture cards are 10. Ace is 1 or 11, player may choose.
- Once the card have been dealt ask the user if they want to twist or stick.
- If the user twists then a new card is dealt and that is added to their total.
- If the player busts (goes over 21) the dealer wins.
- Once the player sticks, the dealer turns over his card and may twist or stick.
- The player or dealer cannot stick below 16.
- If the dealer busts then the player wins.
- If both players stick, the player closest to 21 wins.

National lottery challenge

Difficulty: ✖✖✖✖ A

In the National Lottery, players pick 6 numbers between 1 and 59. Six unique random balls are then output, together with a "bonus ball". Prizes are awarded for matching 2, 3, 4, 5, 5 and the bonus ball or 6 numbers. Write a program to simulate the National Lottery, outputting from 1,000,000 draws how many times the player won each of the prizes. Use your program to prove the odds of winning the jackpot of 6 numbers is 1 : 45,057,474.

London Underground challenge

Difficulty: �✕✕✕ A

The only station on the London Underground that can be formed without using any of the letters in the word mackerel is St John's Wood. This is also true for the words piranha and sturgeon, although for different stations.

For a given list of stations, write a program that takes a word and determines if there is a single station that can be formed without using any of its letters.

Mayan Calendar

Difficulty: ✕✕✕✕ A

In the Mayan calendar there were 20 days (called kins) in a uinal, 18 uinals in a tun, 20 tuns in a katun and 20 katuns in a baktun. We write our dates in the order day, month then year. The Maya wrote their dates in reverse, giving the baktun, katun, tun, uinal then kin.

The Mayan date 13-20-7-16-3 corresponds to 1-1-2000. Write a program which converts between the Mayan calendar and our own.