Python Code Bank

for KS4 Computer Science

POD 7700

BW10.7700

computerscience@zigzageducation.co.uk
zigzageducation.co.uk

Contents

Thank You for Choosing ZigZag Education ii

Teacher Feedback Opportunity iii

Terms and Conditions of Use iv

Teacher's Introduction 1

A4 Handouts 2

- Output 2
- Input 3
- Variables 4
- Casting 6
- Numeric Data Manipulation 7
- Selection: IF 9
- Selection: IF ELSE 10
- Selection: IF ELIF 12
- Operators: Relational 15
- Operators: Boolean 17
- String Manipulation: Length 19
- String Manipulation: Substring 20
- String Manipulation: Case 21
- String Manipulation: Concatenation 22
- String Manipulation: Split 23
- String Manipulation: Find 24
- String Manipulation: Type Check 26
- String Manipulation: ASCII 27
- Iteration: FOR Loop 28
- Iteration: WHILE Loop 30
- Lists 32
- 2D Lists (List of Lists) 34
- List Tools 38
- Reading from a File 40
- Appending to a File 42
- Overwriting a File 44
- Subroutines 46
- Searching 52
- Sort 55
- Random Number Generation 57
- Dictionary 58

Appendix: A5 Booklet

Teacher's Introduction

This resource has been written to provide students with explanations and examples of the core programming techniques available in the Python^3 programming language.

The range and complexity of the techniques and examples covered in this resource make it ideal for KS4 level (it has been produced with GCSE Computer Science specifications in mind) – however it could be used at any key stage where students are learning to program. For example, by familiarising students with the resource during KS3 lessons, they will know how to make use of it at GCSE.

Students can then refer to the syntax, and adapt code for use in their own programs.

Important: if you are intending to use this resource to support students while working on their non-exam assessments (NEA), it is your responsibility to ensure that the support you provide students with is appropriate, including meeting any guidelines set out by your exam board.

The techniques covered have been broken into 33 different topics, each consisting of the following:

1. *Description of the code* – detailing the purpose of the code, and the valid syntax structure needed.
2. *Code in context* – a series of short, generic code snippets showing examples of each technique in use. Each one is summarised in plain English, with comments throughout the code to explain how it works.

Each topic is provided as one or more separate A4 pages, making it easy for you to select the ones you want to hand out to students. Alternatively, an A5 mini-booklet format is provided, allowing to you to hand out the entire code bank to students. A Word version is also provided on disk, allowing you to edit and print the worksheets – including in colour should you want to.

In addition to the paper formats, the code snippets are also provided electronically in the following ways:

1. As 133 individual *PY* files, which students can import into the integrated development environment of their choice, to see how they run, to edit and manipulate, or to incorporate in their own programs.
2. A *HTML* interface includes all of the snippets, and also gives students the ability to edit and run them from within their web browser.

This resource may be used on your school network by copying the files from the CD to a location which is accessible to students.

The CD contains three folders: one containing the code snippets in **PY** format, one with the code snippets in **HTML** format, and one containing a **DOCX** version (for editing/printing from MS Word).

Output

Description of Code

Output allows the user to print data to the screen for the user to read.

print(<data to output>)

This code outputs the data inside the brackets to the screen. This data can be
World' or a variable such as theNumber.

Outputting multiple values requires a **+** symbol between them (see examples

print(<data to output> + <data to output>)

All data being output must be a string, so if a variable stores a number, it need
being output (see Example 3), e.g.:

print(str(<data to output>))

Using **\n** within a print statement forces a new line.

Code in Context

1. The text '*Hello World'* is outputted.

```
# output the text "Hello World" to the scr
print("Hello World \n")
```

2. A variable is used to store a value; which is then outputted.

```
# store 12 in the variable theNumber
theNumber = 12

# output the value in theNumber
print(str(theNumber) + "\n")
```

3. A variable is used to store a value; which is then outputted along with sor

```
# store 12 in the variable theNumber
theNumber = 12

# convert the value in theNumber to a string, output this and the
print(str(theNumber) + " is a number\n")
```

4. A variable is used to store a value; which is then outputted along with sor
before and after it.

```
# store 12 in the variable theNumber
theNumber = 12

# output the value in theNumber as a string, surrounded by two ot
print([illegible]umber " + str(theNumber) + " is my favourite number\n
```

Input

Description of Code

Input allows the user to enter some data that can then be used in the program.

The code:

input()

waits for the user to enter some text and press the return/enter key. The inpu
If you need it to be stored as a number or other data type, you will need to cas

The input can be requested by outputting a message first, using:

input(<data to output>)

Code in Context

1. The program allows data to be input before outputting the same data.

```
# read data from the console and output it
print(input("Type something \n") + "\n")
```

2. The text '*Enter a number*" is output; the inputted data is stored as a string i then output.

```
# display the text "Enter a number", store the value input in the
theNumber = input("Enter a number \n")

# output the content of theNumber
print(str(theNumber) + "\n")
```

3. The message '*Enter a number*' is output; the data input is output to the scr

```
# output "Enter a number", read in a value from the user, output t
print(input("Enter a number \n"))
```

4. The message '*Enter a number*' is output; the data input is cast as an intege **theNumber.**

```
# output "Enter a number", read the value the user inputs, conver
store it in the variable theNumber
theNumber = int(input("Enter a number \n"))

# output the content of theNumber as a string
print(str(theNumber) + "\n")
```

Variables

Description of Code

A variable is a space in memory that stores a piece of data that can change. Y location a name so it can be easily accessed. You can put data in the memory data out of it.

Once you have used a variable with one type of data, it can only have data of t For example, if you put a string into a variable, it can only hold strings.

Data Types

Name	Description	E
Integer	Whole numbers	0, 33, -1
Real	Numbers with decimal parts	2.6, -9.1
Boolean	True or False	True, Fa
String	Characters, including symbols and numbers that do not need to be used in mathematical calculations	“Hello W “22.6”, “

String values are always surrounded by speech marks, or quotes. This tells Py not variable names.

Variable names cannot start with a number or symbol, they cannot have space use reserved words (these are words used by Python, such as if, elif, def, etc.).

Putting data in a variable

```
<variable name> = <data or expression>
```

The = can be read as becomes, so the variable on the left becomes the data or For example, in this code the variable myVariable becomes the number 123.

```
myVariable = 123
```

Getting data from a variable

To access the data in a variable, use its name. For example, to output the con

```
print(<variable name>)
```

Code in Context

1. A variable called **myNumber** has the number stored in it.

```
# myNumber becomes 123
myNumber = 123

# output the value in myNumber
print(str(myNumber) + "\n")
```

2. A variable called **favouriteFilm** as a string has “The Matrix” stored in it.

```
# favouriteFilm becomes "The Matrix"
favouriteFilm = "The Matrix"

# output the value in favouriteFilm
print(favouriteFilm + "\n")
```

3. A variable called **continueFlag** as a Boolean has True stored in it.

```
# continueFlag becomes True
continueFlag = True

# output the value in continueFlag
print(str(continueFlag) + "\n")
```

4. A variable called **myNumber** is given the value of 2.5, which is then output

```
# myNumber becomes 2.5
myNumber = 2.5

# output the value stored in myNumber
print(str(myNumber) + "\n")
```

5. A variable called **cityName** is given the value of “London”, which is then o

```
# cityName becomes "London"
cityName = "London"

# output the data in cityName
print(cityName + "\n")
```

Casting

Description of Code

Some data can be converted to a different data type. For example, the string "1
converted to the integer 123.

The code:

int(<data to convert to an integer>)

converts the data inside the brackets into an integer (whole number).

Instead of int we can also use:

- **str** to convert to a string
- **float** to convert to a decimal number

Code in Context

1. The program reads a string from the user, converts it to an integer, adds 1
   to a string to output.

```
# ask user to enter a number, convert it to integer and store in t
userNumber = int(input("Enter a number \n"))

# add 10 to the value in userNumber
userNumber = userNumber + 10

# convert the number to a string and output it
print("The number is " + str(userNumber) + "\n")
```

2. The program reads two numbers from the user, converts each to a decima
   variables. These two numbers are added together. The total is conve

```
# ask the user to enter the cost, covert it to a decimal and stor
cost1 = float(input("Enter the first cost \n"))

# ask the user to enter the cost, covert it to a decimal and stor
cost2 = float(input("Enter the second cost \n"))

# add together the values in cost1 and cost2, store the result in
total = cost1 + cost2

# convert the value in total to a string and output it
print(str(total) + "\n")
```

3. Read in a number as a string, and store it in **stringValue**. Convert it to a d
   Convert the value in **floatValue** to a whole number and store it in **integerV**

```
# ask the user to input a number, store it as a string in the var
stringValue = input("Enter a number \n")

# convert the value in stringValue to a decimal and store it in t
floatValue = float(stringValue)

# convert the value in floatValue to a whole number and store in t
integerValue = int(floatValue)

# output text and the value in stringValue
print("The string is ", stringValue + "\n")

# output the text, convert the value in floatValue to a string an
print("The decimal is ", str(floatValue) + "\n")

# output the text, convert the value in integerValue to a string a
print("The whole number is ", str(integerValue) + "\n")
```

Numeric Data Manip

Description of Code

Mathematical operations can be performed on numerical data, using either the a variable holding the data.

There are many mathematical operations you can perform; the most common

Symbol	Function	Example	Explan
+	Addition	X = 3 + 4	X would now store 7.
-	Subtraction	X = 5 – 2	X would now store 3.
*	Multiplication	X = 2 * 3	X would now store 6.
/	Division	X = 6 / 3	X would now store 2.
**	Exponential	X = 2 ** 3	X would now store 2 to the powe

%	Modulus	X = 10 % 5
		X = 10 % 4
			This keeps only the remainder of
			10/5 = 2, remainder 0. Therefore
			10/4 = 2.5. 4 * 2 = 8, so there is a
			Therefore 10 % 4 would return 2

//	Division	X = 5 // 3	X stores the integer part of the di 5/3 = 1.666, so 5//3 would return

Code in Context

1. The program stores 10 and 20 in two variables, adds them together and o

```
# store 10 in the variable num1, 20 in the variable num2
num1 = 10
num2 = 20

# add num1 and num2, store the result in the variable total
total = num1 + num2

# output "num1 + num2 = total"
print(str(num1) + " + " + str(num2) + " = " + str(total) + "\n")
```

2. The program stores 10 and 20 in variables, subtracts the value in num2 fr the result.

```
# store 10 in the variable num1, 20 in the variable num2
num1 = 10
num2 = 20

# subtract num2 from num1, store the result in total
total = num1 - num2

# output "num1 - num2 = total"
print(str(num1) +  " - " + str(num2) + " = " + str(total) + "\n")
```

3. The program stores 10 and 20 in variables, multiplies the values together

```
# store 10 in the variable num1, 20 in the variable num2
num1 = 10
num2 = 20

# multiply num1 and num2, store the result in total
total = num1 * num2

# output "num1 * num2 = total"
print(str(num1) + " * " + str(num2) + " = " + str(total) + "\n")
```

4. The program stores 10 and 20 in variables, divides the 10 by 20 and outpu

```
# store 10 in the variable num1, 20 in the variable num2
num1 = 10
num2 = 20

# divide num1 by num2, store the result in total
total = num1 / num2

# output "num1 / num2 = total"
print(str(num1) + " / " + str(num2) + " = " + str(total) + "\n")
```

5. The program stores 10 and 3 in variables, calculates 10^3 and outputs the

```
# store 10 in the variable num1, 3 in the variable num2
num1 = 10
num2 = 3

# raise num1 to the power of num2, store the result in total
total = num1 ** num2

# output "num1 ** num2 = total"
print(str(num1) + " ** " + str(num2) + " = " + str(total) + "\n")
```

6. The program stores 10 and 3 in variables, calculates the modulus division and outputs the result.

```
# store 10 in the variable num1, 3 in the variable num2
num1 = 10
num2 = 3

# calculate num1 MOD num2, and store the result in total
total = num1 % num2

# output "num1 % num2 = total"
print(str(num1) + " % " + str(num2) + " = " + str(total) + "\n")
```

7. The program stores 10 and 3 in variables, calculates the integer division o result.

```
# store 10 in the variable num1, 3 in the variable num2
num1 = 10
num2 = 3

# calculate num1 DIV num2, store the result in total
total = num1 // num2

# output "num1 // num2 = total"
print(str(num1) + " // " + str(num2) + " = " + str(total) + "\n")
```

Selection: I

Description of Code

Selection statements let you run code depending on conditions. The code wil
is true, but will not be run if it is false. There are three levels of IF statement:
ELIF.

IF

```
if <condition>:
   <code to run if condition is true>
```

If the condition is true, then the code within the IF statement will run. If the c
statement is skipped and the program continues below the IF statement.

All code that you want to run within the IF statement needs to be indented at

Code in Context

1. The program outputs *"The number is 10"* if the value in **theNumber** is equa

```
# store 10 in the variable theNumber
theNumber = 10
if theNumber == 10:                   # if the value in theNumber
    print("The number is 10 \n")      # output this message
```

2. The program outputs *"Correct"* if the value in **username** is equal to *"Bob12*

```
# store "Bob123" in the variable username
username = "Bob123"
if username == "Bob123":              # if the value in username
    print("Correct \n")               # output this message
```

3. The program adds 10 to **num1** if the value in **num1** is less than 10.

```
# store 2 in the variable num1
num1 = 2
if num1 < 10:                  # if the value in num1 is less tha
    num1 = num1 + 10           # add 10 to the value in num1, and

# output the value in num1
print(str(num1) + "\n")
```

4. The program subtracts 10 from the value in **num1** if the value in **num1** is

```
# store 2 in the variable num1
num1 = 2

if num1 > 10:                  # if the value in num1 is greater t
    num1 = num1 - 10           # subtract 10 from the value in nu

# output the value in num1
print(str(num1) + "\n")
```

5. The program asks the user to enter a number; if this number is equal to 1

```
# store 10 in the variable num
num = 10

# prompt user input a number and store it as an integer
userNumber = int(input("Enter a number \n"))

if userNumber == 10:           # if the value in userNumber is eq
    print("Correct \n")        # output this message
```

Selection: IF E

Description of Code

Selection statements let you run code depending on conditions. The code wil is true, but will not be run if it is false. There are three levels of IF statement: ELIF.

IF ELSE

```
if <condition>:
    <code to run if condition is true>
else
    <code to run if condition is false>
```

If the condition is true, then the code within the IF statement will run. If the c in the ELSE statement will run.

It is important that the code inside the IF and the ELSE is indented to the sam

Code in Context

1. The program outputs "The number is 10" if the value in **theNumber** is equ outputs "The number is not 10".

```
# the value 10 is stored in the variable theN
theNumber = 10

if theNumber == 10:                    # if the value in theNumbe
    print("The number is 10 \n")       # output this message
else:                                  # if the value in theNumbe
    print("The number is not 10 \n")   # output this message
```

2. The program outputs "Correct" if the value in **username** is equal to "Bob12 "That is incorrect".

```
# store the value Bob123 in the variable username
username = "Bob123"

if username=="Bob123":                 # if the value in username
    print("Correct \n")                # output this message
else:                                  # if the value in username
    print("That is incorrect \n")      # output this message
```

3. The program adds 10 to the value in **num1** if the value in **num1** is less tha from the value in **num1**.

```
# store the value 2 in the variable num1
num1 = 2

if num1 < 10:             # if the value in num1 is less than 10
    num1 = num1 + 10      # add 10 to the value in num1 and stor
else:                     # if the value is num1 is not less tha
    num1 = num1 - 10      # subtract 10 from the value in num1 a
                          # num1

# output the value in num1
print(str(num1) + "\n")
```

4. The program subtracts 10 from the value in **num1** if the value in **num1** is
"Too small".

```
# store the number 2 in the variable num1
num1 = 2

if num1 > 10:                # if the value in num1 is greater than
    num1 = num1 - 10         # subtract 10 from value in num1 and s
else:                        # if the value in num1 is not greater t
    print("Too small \n")   # output this message

# output the value in num1
print(str(num1))
```

5. The program asks the user to input a number. If that value is equal to the
outputs "Correct". If not, it outputs "Incorrect".

```
# store the value 10 in the variable num
num = 10

# ask the user to input a number, store it in the variable userNu
userNumber = int(input("Enter a number \n"))

if userNumber == num:        # if the value in userNumber is equal
    print("Correct \n ")     # output this message
else:                        # if the value in userNumber is not eq
    print("Incorrect \n")    # output this message
```

Selection: IF E

Description of Code

Selection statements let you run code depending on conditions. The code wil is true, but will not be run if it is false. There are three types of IF statement: ELIF.

IF ELIF

The code:

```
if <condition>:
    <code to run if condition is true>
elif <condition>:
    <code to run if this condition is true>
```

If the condition is true, then the code within the IF statement will run. If the c condition will be checked; if this is true, the second set of code will run.

Any number of ELIFs can be added; for example:

```
if <condition> :
    <code to run if condition is true>
elif <condition> :
    <code to run if this condition is true>
elif <condition> :
    <code to run if this condition is true>
elif <condition> :
    <code to run if this condition is true>
```

In this example, if the first condition is false, it will check the second; if this is If one of the conditions is true, then the code within the condition will run and checked.

This can also be combined with an ELSE; for example:

```
if <condition> :
    <code to run if condition is true>
elif <condition> :
    <code to run if this condition is true>
elif <condition> :
    <code to run if this condition is true>
elif <condition> :
    <code to run if this condition is true>
else
    <code to run if none of the conditions are t
```

There can only be one ELSE statement, which is last in the list; this will only r are true.

Code in Context

1. The program compares the value in **guess** to the value in **theNumber**. If t
“Correct”. If they are not equal, but **guess** is less than **theNumber**, it outp
equal, and **guess** is not less than **theNumber**, it outputs “Too large”.

```
# store the value 10 in the variable theNumber
theNumber = 10

# ask the user to input a number, store the input in the variable
guess = int(input("Guess the number \n"))

if guess == theNumber:         # if guess is equal to theNumber
    print("Correct \n")        # output this message
elif guess < theNumber:        # if not, check if guess is less tha
    print("Too small \n")      # output this message
else:                          # if neither if condition is true
    print("Too large \n")      # output this message
```

2. The program asks the user to input a score. If the value in **score** is greate
“Brilliant, well done”. If not, it checks whether the value is greater than o
program outputs “Fab, you did really well”. The program continues checki
If the value does not meet any of the criteria, it outputs “Oh dear, some ex

```
#ask the user to input a score, store this in score as in integer
score = int(input("Enter your score \n"))

if score >= 90:                           # if score is greater t
    print("Brilliant, well done \n")      # output this message
elif score >= 80:                         # if not, check if scor
    print("Fab, you did really well \n")  # output this message
elif score >= 70:                         # if not, check if scor
    print("That was pretty good \n")      # output this message
elif score >= 60:                         # if not, check if scor
    print("Not bad, but I think you can do better \n") # output t
elif score >= 50:                         # if not, check if scor
    print("You got at least half marks, you can improve on that \
elif score >= 40:                         # if not, check if scor
    print("Not quite half marks, need to try harder \n") # output
else:                                     # if none of the previo
    print("Oh dear, some extra work needed here \n") # output thi
```

3. The program asks the user to input a subject. If the value in **subject** is eq
it outputs “Good choice”. If not, it compares it to “Maths”, “French” and “P
any, it outputs “Is that even a subject?”.

```
# ask the user to input their favourite subject,
# and store this in the variable subject as an integer
subject = input("Enter your favourite subject \n")

if subject == "Computer Science":          # if subject is equal
    print("Good choice \n")                # output this message
elif subject == "Maths":                   # if not, check if sub
    print("What's [illegible]98? \n")      # output this message
elif subject == "French":                  # if not, check if sub
    print("Très bien \n")                  # output this message
elif subject == "Physics":                 # if not, check if sub
    print("Why does Earth go around the Sun? \n") # output this m
else:                                      # if none of the previ
    print("Is that even a subject? \n")    #output this message
```

4. The programs tells the user to enter a number. If the number is less than If not, but it is less than 25, it adds 5 to it. If it is not less than 25, it subt The program then outputs the value in **numEntered**.

```
#ask the user to input a number, store this in the variable numEn
numEntered = int(input("Enter a number \n"))

if numEntered < 10:                  # if numEntered is less than 1
    numEntered = numEntered + 10     # then add 10 to the value in
elif numEntered < 25:                # if not, check if numEntered
    numEntered = numEntered + 5      # then add 5 to the value in n
else:                                # if neither condition is true
    numEntered = numEntered - 2      # subtract 2 from the value in
print(str(numEntered) + "\n")        # output the value in numEnter
```

Operators: Relat

Description of Code

Relational operators are used in comparisons; for example, in selection (IF) an

They are used in expressions which return either true or false.

Operator	Description
<	**Less than** *Is the value to the left of the operator less than the value to the right*
>	**Greater than** *Is the value to the left of the operator bigger than the value to the right?*
<=	**Less than or equal to** *Is the value to the left of the operator less than, or equal to, the value to th*
>=	**Greater than or equal to** *Is the value to the left of the operator bigger than, or equal to, the value to*
==	**Equal to** *Is the value to the left of the operator equal to the value to the right?*
!=	**Not equal to** *Is the value to the left of the operator not equal to the value to the right?*

Code in Context

1. The program asks the user to enter two numbers; it then outputs the larg

```
# ask the user to enter two numbers, store them in num1 and num2
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))

if num1 < num2:                  # if num1 is less than num2
    print(str(num2) + "\n")      # output the value in num2
else:                            # if the condition is not met
    print(str(num1) + "\n")      # output the value in num1
```

2. The program asks the user to enter two numbers; it then outputs the sma

```
# ask the user to enter two numbers, store them in num1 and num2
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))

if num1 > num2:                  # if num1 is greater than num2
    print(str(num2) + "\n")      # output the value in num2
else:                            # if the condition is not met
    print(str(num1) + "\n")      # output the value in num1
```

3. The program asks the user to enter two numbers. It outputs the larger of the same, it outputs "Same".

```
# ask the user to enter two numbers, store them in num1 and num2
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))

if num1 > num2:                     # if num1 is greater than num
    print(str(num1) + "\n")         # output the value in num1
elif num1 < num2:                   # if not, check if num1 is le
    print(str(num2) + "\n")         # output the value in num2
else:                               # if neither condition is tru
    print("Same \n")                # output this message
```

4. The program asks the user to enter two numbers. It outputs "Same" if they

```
# ask the user to enter two numbers, store them in num1 and num2
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))

if num1 == num2:                # if num1 is equal to num2
    print("Same \n")            # output this message
else:                           # if the if condition is false
    print("Different \n")       # output this message
```

5. The program asks the user to enter two numbers. It outputs "Different" if or "Same" if they are equal.

```
# ask the user to enter two numbers, store them in num1 and num2
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))

if num1 != num2:                    # if num1 is not equal to num
    print("Different \n")           # output the message
else:                               # if the condition is false
    print("Same \n")                # output this message
```

Operators: Bool

Description of Code

Boolean operators are used in comparisons; for example, in selection (IF) and i

They are used in expressions which return either true or false. AND and OR ta and determine whether the result is true or false.

Operator	Description	
and	**Logical AND** All conditions must be true for the outcome to be true.	2 < 2 < 10 <
or	**Logical OR** At least one condition must be true for the outcome to be true.	2 < 10 < 2 < 10 <
not()	**Logical NOT** If the statement in the brackets is true, return false. If the statement in the brackets is false, return true.	not not

Code in Context

1. The program asks the user to enter four numbers. If the 1st number is less is less than the 4th, it adds together the 1st and 3rd values. If not, it adds t

```
# ask the user to enter four numbers, store them in the variables
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))
num3 = int(input("Enter a third number \n"))
num4 = int(input("Enter a fourth number \n"))

if num1 < num2 and num3 < num4:   # if num1 is less than num2, AN
    total = num1 + num3           # total becomes num1 plus num3
else:                             # otherwise
    total = num2 + num4           # total becomes num2 plus num4
print(str(total) + "\n")          # output the value in the varia
```

2. The program asks the user to enter four numbers. If the 1st number is less number is less than the 4th, it adds together the 1st and 3rd values. If not,

```
# ask the user to enter four numbers, store them in the variables
num1 = int(input("Enter a number \n"))
num2 = int(input("Enter a second number \n"))
num3 = int(input("Enter a third number \n"))
num4 = int(input("Enter a fourth number \n"))

if num1 < num2 or num3 < num4:    # if num1 is less than num2 OR
    total = num1 + num3           # total becomes the num1 plus n
else:                             # otherwise
    total = num2 + num4           # the value in total becomes nu
print(str(total) + "\n")          # output the value in the varia
```

3. The program asks the user to enter two numbers. If both numbers are gre outputs "You passed both". If not, but one of the numbers is greater than "You passed one". If neither is greater than or equal to 60, it outputs "You

```
#ask the user to enter two numbers, store them in the variables m
mark1 = int(input("Enter a number \n"))
mark2 = int(input("Enter a second number \n"))

# if mark1 is greater than to 60, AND mark2 is greater than to 60
if mark1 >= 60 and mark2 >= 60:
    print("You passed both \n")         # output the message

# if mark1 is >= to 60, OR mark2 is >= to 60
elif mark1 >= 60 or mark2 >= 60:
    print("You passed one \n")          # output the message

# if neither condition is true
else:
    print("You didn't pass either \n")     # output the message
```

4. The program asks the user to input a subject. If the value is not "Comput How can that be?".

```
# ask the user to input a subject, store it in the variable subje
subject = input("What is your favourite subject? \n")

if not(subject == "Computer Science"):     # if subject is not eq
    print("What!! How can that be? \n")    # output the message
```

5. The programs asks the user to input a number and stores it in **num1**. It lo is not equal to 10. Within each iteration, it adds 1 to the value in **num1**.

```
# ask the user to input a number, store it in the variable num1 a
num1 = int(input("Enter the first number \n"))

while not(num1 == 10):           # loop, while the value in num1 is
    num1 = num1 + 1              # add 1 to the value in num1
print(str(num1) + "\n")          # output the value in num1
```

6. The program asks the user to enter two numbers. It loops until either of t not true. It counts the number of times it loops, and adds 1 to the value in outputs the number of times it runs.

```
# ask the user to input two numbers, store them in the variables
num1 = int(input("Enter the first number \n"))
num2 = int(input("Enter the second number \n"))

# set count to 0
count = 0

# loop while both (the value in num1 is less than 10)
# AND (the value in num1 is less than the value in num2) are fals
while not(num1 > 10 and num2 > num1):
    num1 = num1 + 1                                  # add 1 to t
    count = count + 1                                # add 1 to t

print(str(count) + "\n")  # output the value in count
```

String Manipulation

Description of Code

We can find out the length of a string, either in a variable or within quotes ("")

The code:

`len(<string>)`

returns the number of characters within the string as an integer.

Code in Context

1. Store “Hello World” in a variable and output the number of characters in t

```
# store "Hello World" in the variable newString
newString = "Hello World \n"

# output the length of the string in newString
print(str(len(newString)) + "\n")
```

2. Count the number of characters in “This is a sentence” and output the resu

```
# store "This is a sentence" in the variable words
words = "This is a sentence \n"

# store the number of characters in words in the variable wordsLe
wordsLength = len(words)

# output text with the value in the variable wordsLength
print("This sentence has " + str(wordsLength) + " characters in it \
```

3. Ask the user to input a colour, then output the numbers from 0 to the num entered.

```
# ask the user to enter a colour, store it in the variable colour
colour = input("Enter your favourite colour \n")

# loop from 0 to the number of characters in the variable colour
for count in range(0, len(colour)):
  # output current value of count
  print(str(count) + "\n")
```

4. Ask the user to input a four-letter word; if it is not four letters, tell them; o word.

```
# ask the user to input a 4-letter word, save it in the variable
userInput = input("[illegible] enter a four letter word \n")

if len(userInput) != 4:                       # if the length
   print("Sorry that's not a 4-letter word \n")   # output the mes

else:                                         # otherwise (the
  print("That was a good word\n")             # output the mes
```

String Manipulation:

Description of Code

You can extract specific characters from within a string; for example, if you ha
"Hello World", you can extract the first five letters and just have "Hello".

Python treats a string as a list of characters, so you can reference them as you

The code:

`<string>[<start index>:<end index>]`
returns the part of the string starting on the first index, ending on the second

Code in Context

1. Store "Hello World" in a variable, then extract and output "Hello" from it.

```
# store "Hello World" in the variable newWords
newWords = "Hello World"

# output the first 5 characters in the variable newWords
print(newWords[0:5] + "\n")
```

2. Output the first 10 characters that the user inputs.

```
# ask the user to input a message, store it in the variable theIn
theInput = input("Type a message \n")

# store the first 10 characters of theInput in the variable extra
extract = theInput[0:10]

# output the text and the value in extract
print("The first 10 characters are " + extract + "\n")
```

3. Output the first half of the characters the user inputs.

```
# ask the user to input a message, store it in the variable theIn
theInput = input("Type a message \n")

# count the number of characters in theInput and divide it by 2 a
# nearest integer, then store it in inputLength
inputLength = int(len(theInput)/2)

# get the characters starting at 0 to inputLength and store in va
extract = theInput[0:inputLength]

# output the text and the value in extract
print("The first half of the message is " + str(extract) + "\n")
```

4. Output each character from a string one character at a time.

```
# ask the user to input a message, store it in the variable theIn
theInput = input("Type a message \n")

# count the number of characters in theInput and store it in inpu
inputLength = len(theInput)

for x in range (0, inputLength):  # loop from 0 to the number of
    print(theInput[x:x+1] + "\n") # output character at position
```

String Manipulatio

Description of Code

A string can be turned into lower case, or into upper case.

The code:

<string>.upper()

will turn each letter in the string to upper case. Characters that are letters

The code:

<string>.lower()

will turn each letter in the string to lower case. Characters that are not letters

Code in Context

1. Output "Hello World" in all lower case, then all upper case.

```
# store "Hello World" in the variable theText
theText = "Hello World"

# convert the contents of theText to upper case, store it in uppe
upperCase = theText.upper()

# convert the contents of theText to lower case, store it in lowe
lowerCase = theText.lower()

# output the contents of lowerCase, and the contents of upperCase
print(lowerCase + " " + upperCase + "\n")
```

2. Convert the first half of a string to lower case and the second half to uppe

```
# ask the user to input a message, store it in inputText
inputText = input("Enter a message \n")

# count the number of characters in inputText, store it in lengthT
lengthText = len(inputText)

# divide the number of letters by 2 and round to an integer, stor
mid = int(lengthText/2)

# convert the first half of inputText to lower case, store in fir
firsthalf = inputText[0:mid].lower()

# convert the second half of inputText to upper case, store in se
secondhalf = inputText[mid:lengthText].upper()

# output the values of firsthalf and secondhalf
print(firsthalf + secondhalf + "\n")
```

String Manipulat
Concatenatio

Description of Code

Concatenation means joining two strings together to become one string.

The code:

<string> + <string>

joins the two strings together to form one string.

Code in Context

1. Join “Hello” and “World” with a space to become “Hello World”.

```
# store Hello in variable named first
# store World in variable named second
first = "Hello"
second = "World"

# store the content of first, a space, then the content of second
message = first + “ ” + second

# output the contents of message
print(message + “\n”)
```

2. Ask the user to enter their first name and surname; then output “Hello” fo
   then su

```
# ask the user to enter their first name and surname
# store the input in variables firstname and surname
firstname = input("Enter your firstname \n")
surname = input("Enter your surname \n")

# output Hello, followed by the user’s name
print("Hello " + firstname + "  " + surname + "\n")
```

3. Ask the user to input a colour and an animal. Concatenate the colour and
   output it in a sentence.

```
# ask the user to input a colour, store in colour
colour = input(“Enter your favourite colour \n”)

# ask the user to input an animal, store in animal
animal = input(“Enter your favourite animal \n”)

# store the content of colour, a space, then the content of anima
final = colour + “ ” + animal

# output the text with the content of final
print(“A ” + final + “ is an interesting animal \n”)
```

String Manipulatio

Description of Code

Split allows a string to be separated into a list of strings, split on a certain cha
For example, you could split a sentence into individual words by splitting on t

The code:

```
<string>.split('<character to split on>')
```

will return a list of strings from the original string.

Code in Context

1. Split “Hello World” by the space and output each word on a new line.

```
# store “Hello World” in a variable called theText
theText = "Hello world"

# split the content of theText by the space, store in newText
newText = theText.split(" ")

# output the first string in the list newText
print(newText[0] + "\n")

# output the second string in the list newText
print(newText[1] + "\n")
```

2. Split a sentence into individual words by the commas and output the 6th w

```
# store the string in the variable sentence
sentence = "This,is,a,comma,separated,sentence \n"

# split the string in sentence by the commas, store as a list of
splitSentence = sentence.split(",")

# output the 6th string in the splitSentence list
print(splitSentence[5] + "\n")
```

3. Split three sentences by the full stop and output each sentence on a new

```
# store the text in the variable sentences
sentences = "This is not just one sentence. It is lots of sentenc
full stop before the next one starts."

# split the string in sentences by the full stops, store as a lis
splitSent = sentences.split(".")

print(splitSent[0] + "\n")    # output the first string in the li
print(splitSent[1] + "\n")    # output the second string
print(splitSent[2] + "\n")    # output the third string
```

4. Split a sentence by the spaces and output each word on a new line.

```
# store the text in the variable words
words = "I don't know how many words are here but I want them all

# split the string in words by the spaces, store as a list of str
splitWords = words.split(" ")

for x in range (0, len(splitWords)):   #  loop from 0 to the numb
splitWords
    print(splitWords[x] + "\n")        #  output the string as po
```

String Manipulatio

Description of Code

Find lets you find out whether a string exists within another string, e.g. if there 'astronaut'. It returns the position of the string if it exists or -1 if not.

The code:

`<string>.find('<character or string to find>')`

will return the [illegible] index of the first occurrence of the character or string it does not exist

Code in Context

1. Output the character position of the letter “r” in the word “Purple”.

```
# store “Purple” in the variable myWord
myWord = "Purple"

# find the position of "r" in the string in position
position = myWord.find("r")

# output the value in position
print(str(position) + "\n")
```

2. Output the character position in a sentence where the word “specific” star

```
# store [illegible] in the variable words
words [illegible] to find out if a specific word is in this list of

# find the position of "specific" in the string words
findWord = words.find("specific")

# output the character number where specific starts
print(str(findWord) + "\n")
```

3. Output all of the text in a string starting from the word “Hello”.

```
# store the string in sentence
sentence = "I only want to output what is after this Hello World"

# find the starting character position of "Hello" in sentence, st
findWord = sentence.find("Hello")

# count the number of characters in the string in sentence, store
length = len(sentence)

# if the character position is -1 (it cannot be found)
if findWord == -1:
    print("Could not find Hello \n")  # output the text
else:                               # otherwise print the phras
    print(sentence[findWord: length] + "\n")
```

4. Ask the user to input text. If the word “and” is input, output “Found it”; ot

```
# ask the user to input text, store it in the variable named sent
sentence = input("Enter lots of words \n")

# find the starting character of "and" in sentence, store the pos
findWord = sentence.find("and")

if findWord != -1:          # if the value in findWord is not equ
    print("Found it \n")    # output the text
else:                       # otherwise
    print("Not there \n")   # output the text
```

5. Ask the user to input text without punctuation. If there is a “!”, or a “,” or a
output “I said NO punctuation”

```
# ask the user to input text, store it in the variable named word
words = input("Enter some text, no punctuation \n")

# if a ! , . or ? are found in words then output the text
if (words.find("!") != -1 or words.find(",") != -1 or words.find(
words.find("?") != -1):
    print("I said NO punctuation \n")
```

String Manipulation: T

Description of Code

You can find out whether a string is a specific data type; for example, whether whether it is all lower case or whether it is all upper case.

The code:

```
<string>.isnumeric()
```

returns true if it is numeric, and false if it is not.

The **isnumeric()** can be replaced with isUpper() to check whether it is a check whether it is all in lower case or isalpha() to check whether it is all a

Code in Context

1. Ask the user to input characters. If it's all letters, output "It's all letters wi numbers, output "It's all numbers"; if neither is true, output "It's a mixture"

```
# ask the user to input characters, store in textInput
textInput = input("Enter characters \n")

# if the characters in textInput are all letters
if textInput.isalpha() == True:
    print("It's all letters with no spaces \n")

# if not, check if all the characters are numbers
elif textInput.isnumeric() == True:
    print("It's all numbers \n")

# if neither of the above conditions are true
else:
    print("It's a mixture \n")
```

2. Ask the user to input characters. If all the letters are in upper case, output "N letters are in lower case, output "Thanks for not shouting". If neither is true,

```
# ask the user to input characters, store in textInput
textInput = input("Enter characters \n")

if textInput.isupper() == True:       # if the characters in tex
    print("No need to shout \n")      # output the message
elif textInput.islower() == True:     # if not, check if all the
    print("Thanks for not shouting \n")  # output the message
else:                                  # if there is a combinat
    print("That's quite a mixture \n")   # output the message
```

3. Keep asking the user to enter their first name until they start it with a cap "Thanks" followed by their name,

```
# set variable takeInput to true
takeInput = True
while takeInput == True:
    # input users name, store in theText
    theText = input("Enter your first name \n")
    # if first character in takeInput is lower case output me
    if theText[0].islower() == True:
        print("Don't forget to start with a capital letter \n")
    else:                             # otherwise
        takeInput = False             # set takeInput to be False to

print("Thanks " + theText + "\n")  # output "Thanks" and the stri
```

String Manipulatio

Description of Code

A character can be turned into its ASCII code, and an ASCII code can be turned

The code:

`ord(<single character>)`

returns the ASCII number of the character.

The code:

`chr(<ASCII number>)`

returns the character of the ASCII number.

Code in Context

1. Output the ASCII value of "?".

```
# store ? in the variable letter
letter = "?"

# output the ASCII value of the character in letter
print(str(ord(letter)) + "\n")
```

2. Output the ASCII value of a character the user inputs.

```
# ask the user to enter a character and store the input in letter
letter = input("Enter a character \n")

# output the ASCII value of the character in letter
print(str(ord(letter) + "\n")
```

3. Output the character for the number the user inputs.

```
# ask the user to enter a number and store the input in numInput
numInput = int(input("Enter a number \n"))

# output the character for the ASCII value input
print(chr(numInput) + "\n")
```

4. Ask the user to input a sentence. Output the ASCII value of each characte

```
# ask the user to enter a sentence, store the input
letterInput = input("Enter a sentence \n")

for x in range(0, len(letterInput)):   # loop from 0 to the leng
    letterNum = ord(letterInput[x:x+1]) # turn character number x
    print(str(letterNum) + "\n")        # output the value in let
```

Iteration: FOR I

Description of Code

A FOR loop is a count-controlled loop; you need to know how many times it wi

```
for <variable> in range(<start value>, <end valu
    <statements to repeat>
```

The variable acts as a counter; it is initialised at the start value, the program th at the end of the loop it increases the variable value by 1.

The program then moves back to the FOR and compares the value in the coun values. If it is still within these bounds, it runs the statements again and incre to the start. This keeps on repeating until the value in the counter is outside th

For example:

```
for counter in range(0, 3):
    print(counter)
```

- This code will start by initialising counter to 0. It runs the code inside the lo counter, 0). It increases counter to 1.
- It goes back to the FOR statement and checks whether counter is between 0 inside the loop (outputs 1). It increases counter to 2.
- It goes back to the FOR statement and checks whether counter is between 0 inside the loop (outputs 2). It increases counter to 3.
- It goes back to the FOR statement and checks whether counter is between 0 code in the loop and continues with the program.

Adjusting the Increment

The FOR statement does not have to increase by 1 each time; you can set the v

```
for <variable> in range(<start value>, <end value>, <
     <statements to repeat>
```

Going down. The following code would start counter at 3, and then decrease th time through.

```
for counter in range(3, 0, -1):
    print(counter)
```

Decimal steps. The following code would start counter at 20, and then increas each time through.

```
for counter in range(20, 50, 5):
    print(counter)
```

Code in Context

1. Output the numbers 0, 1, 2, 3.

```
# set counter to start at 0, increase by 1 each time, loop until o
for counter in range(0,4):
    # output the value in counter
    print(str(counter) + "\n")
```

2. Display the times table (up to 12 times the number) for a number the use

```
# ask the user to enter a number, take the input, cast to an inte
limit = int(input("Enter the number of the times table you want to

# set counter to start at 0, increase by 1 each time
# loop until it is outside the range 0-12
for counter in range (0, 13):
    # multiply the value in counter by the value in limit, store i
    result = counter * limit
    #output the message
    print(str(counter) + "*" + str(limit) + "=" + str(result) + "
```

3. Ask the user to input a number; output that many "*"s on the same line.

```
# ask the user to enter a number, cast as an integer and store in
userInput = int(input("Enter the number of *s you want displayed 
message = ""                  # set message to be the empty strin
for x in range(0, userInput): # loop until x is outside the range
    message = message + "*"   # concatenate a "*" to the end of mes
print(message + "\n")         # output the value in message
```

4. Output a countdown from 10 to 1 then display "Blast Off!".

```
# loop from 10 to 1, setting the step to decrease count by 1 each
for count in range (10, 0, -1):
    # output the value in count
    print(str(count) + "\n")

# output the message
print("Blast off! \n")
```

5. Output alternate numbers from 0 to the number the user inputs.

```
# ask the user to input the stop value, cast as an integer and sto
stopValue = int(input("Enter the number to stop at \n"))

# loop from 0 to stopValue, increasing count by 2 each iteration
for count in range (0, stopValue, 2):
    # output the value in count
    print(str(count) + "\n")
```

Iteration: WHILE

Description of Code

A WHILE loop is a condition-controlled loop; it is usually used when you do no times the loop will run, although it can also be used as a count-controlled loo

It loops (and continues looping) while a condition is True. When the condition False, it stops looping.

```
while <condition>:
      <statements to repeat>
```

Code in Context

1. Output the numbers from 0 to 10.

```
# store 0 in counter
counter = 0

# loop while counter is less than or equal to 10
while counter <= 10:
    # output the value in counter
    print(str(counter) + "\n")
    # add 1 to the value in counter
    counter = counter + 1
```

2. Loop asking for input while the user enters "Y".

```
# store "Y" in inputValue
inputValue = "Y"

# loop while the value in inputValue as an upper case is equal to
while inputValue.upper() == "Y":
    # ask the user to enter Y or N, read the input and store it i
    inputValue = input("Enter Y to continue or N to stop \n")

# output the message
print("Ok, we've stopped \n")
```

3. Generate and output random numbers between 0 and 100 until a number

```
# import the random module to generate random numbers
import random

# store 0 in randomNumber
randomNumber = 0

# loop while the value in randomNumber is less than or equal to 5
while randomNumber <= 5
    # generate a random number between 0 and 100, store in random
    randomNumber = random.randint(0, 100)

    # output the value in randomNumber
    print(str(randomNumber) + "\n")
```

4. Ask the user to input numbers until they do not want to continue. Count h
Calculate the mean average of all the numbers entered. Output the larges
smallest number entered.

```
userContinue = "Y"     # store "Y" in userContinue
largest = 0            # store 0 in largest
smallest = 9999        # store 9999 in smallest
total = 0              # store 0 in total
counter = 0            # store 0 in counter

# loop while the value in userContinue is one of the options
while userContinue == "y" or userContinue == "Y" or userContinue
== "yes":

    # ask the user to input a message, read the input and store i
    userNum = int(input("Enter the first number \n"))

    # add userNum to the running total
    total = total + userNum

    # add 1 to the value in counter
    counter = counter + 1

    # check if the input is the largest value so far
    if userNum > largest:    # if the value in userNum is larger
        largest = userNum    # store the value in userNum in larg

    # check if the input is the smallest value so far
    if userNum < smallest:   # if the value userNum is smaller th
        smallest = userNum   # store the value in userNum in smal

    # output "Continue?", take the user input and store in userCo
    userContinue = input("Continue? \n")

# output the number of numbers entered
print("You entered " + str(counter) + " numbers \n")

# output the largest number
print("The largest number is " + str(largest) + "\n")

# output the smallest number
print("The smallest number is " + str(smallest) + "\n")

# calculate and output the average of all the numbers input
print("The mean average of the numbers is " + str(total/counter) +
```

Lists

Description of Code

A list is a structure that can store many pieces of data, unlike a variable, which piece of data. A list has one identifier (name) and then a number of spaces (ca pieces of data can be put in.

A list can be visualised as a table; for example:

Index	0	1	2	
Data	"Red"	"Blue"	"Orange"	

Lists start counting at 0. This list has five elements: index 1 is Blue, index 3 is A list needs to be declared, either as blank list (no data) or with data.

Blank list

```
<identifier> = []
```

For example:

```
colours = []
```

With data

```
<identifier> = [<data>,<data>]
```

For example:

```
colours = ["Blue", "Red", "Green"]
```

Adding data

The append method allows new data to be added to the end of a list; for example, no data, then to add a piece of data it needs to be appended. Similarly, if a list is d fourth is needed, append adds the new piece of data.

```
<identifier>.append(<data>)
```

For example:

```
colours.append("Grey")
```

Accessing data

Data in a specific element can be accessed:

```
<identifier>[<index>]
```

For example, to access the data in colours in index 1:

```
colours[1]
```

Looping through array elements

A FOR loop can be used to access each element in the array in turn:

```
for <identifier> in <arrayName>:
    <identifier>
```

For example, to access each item in the array colours and output that item:

```
for eachItem in colours:
    print(eachItem)
```

Code in Context

1. Store five colours in a list and output each element in the array.

```
# declare a list named colours with 5 elements
colours = ["Red", "Blue", "Orange", "Yellow", "Purple"]

# loop from the first element (0) to the last element (4)
for x in range (0, 5):
    # output the data in the list at position
    print(colours[x] + "\n")
```

2. Ask the user to input five colours; store each one in the array, and then ou

```
# declare an empty array named colours
colours = []

# loop 5 times
for count in range (0, 5):
    # ask the user to input a colour, append the input to the end
    colours.append(input("Enter a colour \n"))

# loop through each element in the array
for eachColour in colours:
    # output the data in that array element
    print(eachColour + "\n")
```

3. Store the months of the year in an array. Ask the user to input a month nu of that month.

```
# store the months in an array named months
months = ["January", "February", "March", "April", "May", "June",
"September", "October", "November", "December"]

# ask the user to input the month, read the input, and store in u
userInput = int(input("Enter the month number \n")) - 1 # subtrac

# output the data element in the array index entered
print(months[userInput] + "\n")
```

4. Take 10 numbers from the user and store in the array userNumbers. Outp element the user inputs.

```
# declare an empty array named numbers
numbers=[]

for count in range (0, 10):
    # input a number, store as an integer and append to the array
    numbers.append(int(input("Enter a number \n")))

# ask the user which number they want to check
# subtract 1 to get the array element to check
userNumber = int(input("Which number do you want to check? \n")) -

# output the number in the array element
print(numbers[userNumber]) + "\n")
```

2D Lists (List of

Description of Code

A 2D list is actually a list of lists. Each item in the list is a list of data items. F
[[0, 1, 2], [3, 4, 5], [6, 7, 8]] This list has three items and each item is a list. Th
[0, 1, 2], the second list of data is [3, 4 ,5], etc.

It can be visualised as a table:

Ind	0	1	2
0	"Red"	"Blue"	"Orange"
1	"Yellow"	"Red"	"Blue"
2	"Red"	"Yellow"	"Purple"

The list now has two indices.

Declaring an empty list of lists
To declare an empty list of lists, the first (outer) list needs to be initialised as havin
value, but, as an example, the number 0 will be used here.

```
<identifier>=[[0]* <first list> for i in range(<
```

In this example, the code will create five lists, with three items in each: [[0, 0, 0], [0,

```
colours=[[0]* 3 for i in range(5)
```

Adding data
Once you have initialised the list of list items, you can add data using the indi

```
<identifier>[<index>][<index>] = <data>
```

In this example, the code will add 25 to the first list, in position 1. The colours arra
[0, 0, 0], [0, 0, 0], [0, 0, 0]].

```
colours[0][1] = 25
```

Accessing Data
Data in a specific list element can be accessed:

```
<identifier>[<index>][<index>]
```

To access the data in colours in position 1, 2:

```
colours[1][2]
```

Code in Context

1. Store shades of four colours in a list; the first 'column' is shades of red, th
   Ask the user to input a colour, and then output the three shades stored fo

```
# store shades of colours in the array, each colour is a sublist
colours = [["Cherry", "Pink", "Magenta"],["Powder Blue", "Aquamar
"Gold", "Cream"],["Avocado","Lime","Olive"]]

# ask the user to enter a colour, convert to upper case and store
userColour = input("Enter a colour: Red, Blue, Yellow or Green \n

if userColour == "RED":                # if the user entered red
    listNumber = 0                     # set the list number to 0
elif userColour == "BLUE":             # if the user entered blue
    listNumber = 1                     # set the list number to 1
elif userColour == "YELLOW":           # if the user entered yellow
    listNumber = 2                     # set the list number to 2
else:                                  # otherwise
    listNumber = 3                     # set the list number to 3

# print the three shades for the colour entered
# loop from 0 to 2
for count in range (0, 3):
    # output the shade at position listNumber and count
    print(colours[listNumber][count] + "\n")
```

2. The list stores 10 players, and five scores for each player. Ask the user to
   each round. Ask the user which player and round they would like to view

```
#declare a list of 10 elements, with 5 sub elements in each list
numbers = [[0] * 5 for i in range(10)]

# loop through each round
for sublist in range(0, 5):
    # loop through each player
    for list1 in range(0, 10):
        # output the player and round, read the input and store i
        numbers[list1][sublist] = int(input("Enter the score for
" round " + str(sublist+1)) + "\n")

# read the player number, cast as integer, store in player
player = int(input("Enter the player number \n"))

# read the round number, cast as integer, store in roundNumber
roundNumber = int(input("Enter the round number \n"))

# output the player number and round entered, and the score for t
print("Player " + str(player) + " scored " + str(numbers[player -
in round " + str(roundNumber) + "\n")
```

3. Store random numbers between 0 and 100 in each element of the list nu
'column' they would like the average calculated, work out the mean avera

```
import random

# declare a list of 4 elements, with 5 elements in each list
numbers = [[0] * 5 for i in range(4)]

# insert a random number into each list element
for column in range(0, 4):
    # loop through each sub element
    for row in range(0, 5):
        # generate a random number between 0 and 100, store in nu
        numbers[column][row] = random.randint(0, 100)

# ask the user which column they would like to calculate the aver
# (subtract 1 to give the index, store in userColumn)
userColumn = int(input("Which column would you like the average f
1

# calculate the average
average = 0                                      # initialise av
for row in range(0, 5):                          # loop through
    average = average + numbers[userColumn][row] # add the numbe

average = average / 4                            # actually comp
print(str(average) + "\n")                       # output the av
```

4. Store the row and column number of each element, in that element. Outp
as a grid.

```
#declare a list of 9 elements with 9 elements in each list
numbers = [[0] * 9 for i in range(9)]

# insert the column and row number in each element
# first loop through each column
for column in range(0, 9):
    # also loop through each row
    for row in range(0, 9):
        # concatenate the column and row, store in numbers in ele
        numbers[column][row] = str(column) + str(row)

# loop through each column
for column in range(0, 9):
    # loop through each row
    for row in range(0, 9):
        # output the value in numbers as position [column][row] w
        print(str(numbers[column][row]) + "|", end="") # stop eac
line
    # force a new line in output
    print("\n")
```

5. Generate 100 random numbers between 1 and 1,000. Ask the user what n
Search the array for that number, and, if it finds it, output the array eleme

```
import random

# declare an empty list named randomNumberList
randomNumberList = []

# loop from 0 to 99
for x in range (0, 100):
    # generate a random number between 0 and 100 and append it to
    randomNumberList.append(random.randint(0, 100))

# ask the user to input a number to find, cast as integer and sto
userInput = int(input(("What number would you like to find? \n"))

# loop through each element in the array, tracking the index
for index in range(0, len(randomNumberList)-1):          # loop f
    if randomNumberList[index] == userInput:             # if the
        print("Found in position" + str(index) + "\n")   # output
        break                                            # break
```

6. Generate 100 random numbers between 1 and 100. Add together all the n
the total.

```
import random

# declare an empty list named randomNumberList
randomNumberList = []

# loop from 0 to 99
for x in range (0, 100):
    # generate a random number between 0 and 100 and append it to
    randomNumberList.append(random.randint(0, 100))

# initialise total var to 0
total = 0

# loop through each element in the array, tracking the index
for item in randomNumberList:
    # add the item to the total
    total = total + item

# output the total
print(str(total) + "\n")
```

List Tools

Description of Code

There are a number of inbuilt methods that can be used on lists.

Length
Returns the number of elements in a list.

len(<arr list as identifier>)

Max
Returns the largest value in the list (e.g. numerically or alphabetically).

max(<list identifier>)

Min
Returns the smallest value in the list (numerically or alphabetically).

min(<list identifier>)

Count
Returns the number of times a piece of data is in the list.

<list identifier>.count(<data>)

Remove
Deletes an item from a list.

<list identifier>.remove(<data>)

Sort
Sorts the contents of a list into ascending order.

<list identifier>.sort()

Reverse
Reverses the order of the elements in the list.

<list identifier>.reverse(<data>)

Code in Context

1. Input 10 numbers into a list. Sort the list into ascending order and outpu

```
# declare an empty list
values = []

# add 10 numbers to the list
for count in range(0, 10):                                # loop from 0
    values.append(int(input("Enter a number \n"))) # take a numbe
values

# sort the data in values into ascending order
values.sort()
# output all the data in values
print(values)
```

2. Ask the user to enter five colours. Sort the colours into descending order

```
# declare an empty list
values = []

# output the message
print("Enter five colours \n")

# take 5 strings as input
for x in range (0, 5):      # loop from 0 to 4
    values.append(input())  # add the input to values

# sort the data in values into ascending order
values.sort()
# reverse the order of the items in values
values.reverse()

# output each item on the same line
for item in values:           # loop through each item in values
    print(item + " ", end="") # output the item and a space, do n
```

3. Ask the user to enter numbers until they do not enter "Y" to continue. Ou entered, the smallest and largest numbers, and how many times they ente

```
# declare an empty array
numbers = []

# set userContinue to True
userContinue = True

# loop while userContinue stores True
while userContinue == True:
    # append input to numbers
    numbers.append(int(input("Enter a number \n")))

    # ask the user to enter Y to continue, if they do not enter Y
    if (input("Enter Y to continue \n").upper() != "Y"):
        # set userContinue to False
        userContinue = False

# output the number of items entered
print("You entered " + str(len(numbers)) + " values \n")
# output the smallest item entered
print("The smallest number was " + str(min(numbers)) + "\n")
# output the largest item entered
print("The largest number was " + str(max(numbers)) + "\n")
# output the number of times 13 was entered
print("You entered the number 13 " + str(numbers.count(13)) + " t
```

4 Ask the user which colour they would like to remove from a list and delet

```
# store colours in the list colours
colours = ["Cherry", "Pink", "[illegible]", "Powder Blue", "Aquamarine
"Gold", "Cream","Avocado", "[illegible]", "Olive"]
# output the contents of the list colours
print(colours)
# read input from the user, store in toRemove
toRemove = input("Which colour would you like to remove \n")
#delete the item input from the list
colours.remove(toRemove)
#output the contents of the list colours
print(colours)
```

Reading from a

Description of Code

A text file lets you store data external to the program file; this means the data another program, or by the same program when it starts running again. If you program, e.g. in an array, it will disappear when the program stops running.

To read the data from a file, you first need to make sure the text file is in a su in the same folder as the executable file (.exe) for the program. Then you only the folders [illegible].

The following code opens a file to be read:

```
<identifier> = open(<file name>, "r")
```

For example:

```
dataFile = open("textFile.txt", "r")
```

Reading all the data in a file

If you want to read all the text from a file, you can use the read function.

```
<identifier>.read()
```

For example:

```
dataFile.read()
```

Reading each line in a file in turn

If you want to read each line separately, you can use a FOR or loop to loop through each line in a file.

```
for <identifier1> in <identifier2>:
    print(<identifier1>)
```

For example:

```
for eachLine in dataFile:
    print(eachLine)
```

Closing the file

As soon as you have finished working with the text file, make sure it is closed.

```
<identifier>.close()
```

Code in Context

1. Output each line in the file “data file 1.txt”.

```
fileName = "data file 1.txt"   # store the filename in fileName
dataFile = open(fileName, 'r') # open the file fileName in read m
theData = dataFile.read()      # read all the data from dataFile
dataFile.close()               # close the text file
print(theData)                 # output the contents of theData
```

2. Ask the user which line in “data file 2.txt” they want to output, then outpu

```
# read the line number the user wants to output, store in userInp
userInput = int(input("Which line would you like to read? \n"))

fileName = "data file 2.txt"   # store the filename in fileName
dataFile = open(fileName, 'r') # open the file fileName in read m
counter = 1                    # store 1 in counter, this will co

# loop through each line in the file until it reaches the one wan
for eachLine in dataFile:      # loop through each line in dataFi
    if counter == userInput:   # if the value in counter is equal
        print(eachLine + "\n") # output that line
    counter = counter + 1      # increment counter

dataFile.close()               # close the text file
```

3. Read the data from a file into an array, and print it.

```
fileName = "data file 2.txt"   # store the filename in fileName
dataFile = open(fileName, 'r') # open the file fileName in read m
counter = 1                    # store 1 in counter, this will co
myList = []                    # create an array

# loop through each line in the file until it reaches the one wan
for eachLine in dataFile:      # loop through each line in dataFi
    myList.append(eachLine)    # add line to array

dataFile.close()               # close the text file
print(myList)                  # print contents of array
```

4. Read the data in a file into a list. Output the three shades of a colour the

```
fileName = "data file 3.txt"   # store the filename in fileName
dataFile = open(fileName, 'r') # open the file fileName in read m
theData = dataFile.read()      # read the data in the text file i
dataFile.close()               # close the text file

# split the data read by the spaces
colours = theData.split(" ")   # split the string theData by " "

# read the colour the user wants to output
userColour = input("Enter a colour you want to output the shades

#loop through the array to find the colour
for count in range(0, len(colours)):  # for each element in colou
    if colours[count] == userColour: # if the element in colours
        for x in range (0, 3):        # loop 3 times
            print(colours[count + x]) # output the shade in colou
```

Appending to a

Description of Code

'Appending' means 'adding to the end of'. Appending to a file means the data
end of the data that is already in the file, instead of overwriting it. The data is
line as the text that is in the file. If the data needs to be on a new line, then \n

The following code opens a file to be appended to:

<identifier> = open(<file name>, "a")

For example:

dataFile = open("textFile.txt", "a")

To append the data, use writelines():

<identifier>.writelines(<data to write>)

For example:

dataFile.writelines("Bob")

Code in Context

1. Write the number 11 to the end of the text file.

```
fileName = "data file 1.txt"   # store the file name of the text

dataFile = open(fileName, "a") # open fileName in append mode as

dataFile.writelines("11")      # append 11 to the text file

dataFile.close()               # close the text file

print("Data written \n")       # output the text
```

2. Write the number 11 to the end of the text file on a new line.

```
fileName = "data file 1.txt"   # store the file name of the text

dataFile = open(fileName, "a") # open fileName in append mode as

dataFile.writelines("\n")      # move to a new line in the text f
dataFile.writelines("11")      # write a new line to the text fil

dataFile.close()               # close the text file

print("Data written \n")       # output the text
```

3. Add the strings "White" and "Silver" to the end of the text file.

```
fileName = "data file 2.txt"   # store the file name of the text

dataFile = open(fileName, "a") # open fileName in append mode as

dataFile.writelines("\n")      # move to a new line in the text f
dataFile.writelines("White")   # write a new line to the text fil
dataFile.writelines("\n")      # move to a new line in the text f
dataFile.writelines("Silver")  # write a new line to the text fil

dataFile.close()               # close the text file

print("Data written \n")       # output the text
```

4. Ask the user to input 10 numbers and append these to the end of the text

```
fileName = "data file 1.txt"      # store the file name of the te

dataFile = open(fileName, "a")    # open fileName in append mode

print("Enter ten numbers \n")     # ask the user to input 10 numb

for x in range (0, 10):           # loop 0 to 10
    dataFile.writelines("\n")     # append a new line to the file
    dataFile.writelines(input())  # read the input, append it to

dataFile.close()                  # close the text file

print("Data written \n")          # output the text
```

5. Input a colour and two shades of that colour. Append them to a text file.

```
# store the file name of the text file
fileName = "data file 3.txt"

# open fileName in append mode as dataFile
dataFile = open(fileName, "a")

# tell the user to input a colour
userInput = input("Enter a colour other than red, yellow, blue or

# append a new line to the file
dataFile.writelines("\n")

# append the user input to the file
dataFile.writelines(userInput)

for x in range (0, 2):
    # append a new line to the file
    dataFile.writelines("\n")
    # ask the user to input a shade of the colour they entered
    dataFile.writelines(input("Input a shade of " + userInput + "

# close the text file
dataFile.close()

# output the text
print("Data written \n")
```

Overwriting a

Description of Code

Overwriting means that the data in the file will be deleted, and then you can a
to write.

The following code opens a file to be overwritten:

<identifier> = open(<file name>, "w")

For example:

dataFile = open("textFile.txt", "w")

To write the data, use writelines():

<identifier>.writelines(<data to write>)

For example:

dataFile.writelines("Bob")

If the file does not exist in the location given, it will be created for you.

Code in Context

1. Write "Hello World" to a text file.

```
fileName = "file1.txt"                   # store the file name in file
fileData = open(fileName,"w")            # open the file fileName in w
fileData.writelines("Hello World")       # write the string to the fil
fileData.close()                         # close the file
print("Hello World")                     # output the string
```

2. Write a user's first name and surname to a text file.

```
#store the file name in fileName
fileName = "userData.txt"

#read the user's first name and surname and store in variables fi
firstName = input("Enter your first name \n")
surname = input("Enter your surname \n")

# open fileName to write, as theFile
theFile = open(fileName,"w")

# write the value in firstName to the text file
theFile.writelines(firstName)

# write a new line to the text file
theFile.writelines("\n")

# write the value in surname to the text file
theFile.writelines(surname)

# close the text file
theFile.close()

print("Data written") # output the string
```

3. Ask the user how many numbers they want to enter. Let them enter this
   them to a text file separated by commas.

```
# store the file name as fileName
fileName = "numbers.txt"

# read the number of numbers the user wants to enter, store in us
userInput = int(input("How many numbers would you like to enter?

# open the file fileName in write mode, as newFile
newFile = open(fileName, "w")

# loop from 0 to userInput
for counter in range(0, userInput):
    # ask the user to enter a number, store as number
    number = input("Enter number " + str(counter + 1) + "\n")
    # write number to the file
    newFile.writelines(number + ",")

# close the file
newFile.close()

# output the string
print("Data written")
```

4. Store the data in an array in a text file.

```
#store the file name as fileName
fileName = "arrayData.txt"

#store colours in the array theData
theData=["Red","Crimson","Pink","Blue","Navy","Azure"]

#open the file in write mode as newFile
newFile = open(fileName, "w")

for item in theData:             # loop through each item in th
    newFile.writelines(item + ",") # write the item and a comma t

# close the text file
newFile.close()

# output the string
print("Data written")
```

Subroutines

Description of Code

A subroutine is an independent piece of code, with its own identifier, that can other places within the program.

Python's subroutines can return a value to the program that called it, but it do can send data (parameters) to the procedure to be used within it.

The return keyword returns a value if needed.

Declaring a subroutine
A subroutine is declared using the code:

```
def <identifier> (<parameters>):
     <subroutine code>
     return <identifier>;
```

Example without parameters, returning a value:

```
def myFunction():
     <subroutine code>
     return "Hello";
```

Example with parameters, returning a value:

```
def myFunction(number):
     <subroutine code>
     Return number;
```

Example not returning a value:

```
def myFunction(value):
     <subroutine code>
```

Calling a subroutine
A subroutine is called using its identifier (with any required parameters in brac subroutine returns a value, then the value will replace the subroutine call, so s happen to this value, e.g. it is stored in a variable, or output.

Example without parameters:

```
myString = mySubroutine()
```

Example with parameters:

```
print(mySubroutine(10))
```

If the subroutine does not return a value, then it can be called using its name.

Example without parameters:

```
mySubroutine()
```

Example with parameters:

```
mySubroutine(10)
```

Program structure
Subroutines need to appear at the start of a Python program. All the code wit indented to the same level. The main program is below the subroutines and i

Code in Context

1. Call a subroutine to ask the user to enter their name. Then call a subrout age.

```
#declare a subroutine named firstOutputs, with no parameters
def firstOutputs():
    name = input("What is your name? \n") # read the input, store
    print("Hello " + name + "\n")          # output the name
    # finish the subroutine, return control to the main program w

#declare a subroutine named secondOutputs, with no parameters
def secondOutputs():
    age = int(input("How old are you in years? \n")) # read the i
    print("You are " + str(age) + " years old \n")   # output the
    # finish the subroutine, return control to the main program w


# the main program starts here
firstOutputs()  # call the subroutine firstOutputs
secondOutputs() # call the subroutine secondOutputs
```

2. Ask the user to input numbers until they say "NO". Output if each numbe equal to) 10.

```
# declare a subroutine named outputValue, with one parameter call
# that does not return a value
def outputValue(number):
    # if the value in the parameter number is greater than 10, ou
    if number > 10:
        print("Greater than 10 \n")
    # if the value in the parameter number is less than 10 output
    elif number < 10:
        print("Less than 10 \n")
    # if neither is true output "It is 10"
    else:
        print("It is 10 \n")

# the main program starts here

# store yes in userInput
userInput = "YES"

#loop while the upper-case value of userInput does not equal no
while(userInput.upper() != "NO"):
    # read the user input, store in theNumber
    theNumber = int(input("Enter a number \n"))
    # call the subroutine outputValue, sending theNumber as the p
    outputValue(theNumber)
    # read the user input, store in userInput
    userInput = input("Again? \n")
```

3. Let the user play a short text game: they choose from options and the sto
   their choice.

```
# declare a subroutine called leftDoor
# it takes no parameters and returns no value
def leftDoor():

    #output the story
    print(“You have gone through the first door \n”)
    print(“You are in a room with no doors but there is a window
\n")
    print(“Would you like to go back, or use the brick on the win

    # read the user input and store in userInput
    userInput = input(“Enter Back or Window \n”)

    # if value in userInput is Back
    if userInput.upper() == “BACK”:
        # call the main subroutine
        startStory()

    # otherwise
    else:
        # call the brickWindow subroutine
        brickWindow()

# declare a subroutine called rightDoor
# it takes no parameters and returns no value
def rightDoor():
    print("There is a door in front of you and a trapdoor \n")
    print("Would you like to go through the door, the trapdoor or

    # read the user input and store in userInput
    userInput = input(“Enter Door, Trapdoor or Back \n")

    if userInput.upper() == “BACK”:   # if userInput is equal to
        startStory()                  # call the main subroutine
    elif userInput.upper() == “DOOR”: # if it equal to door
        rightDoor()                   # call the rightDoor proced
    else:                             # if neither if is true
        trapdoor()                    # call the trapdoor subrout

# declare a subroutine called brickWindow
# it takes no parameters and returns no value
def brickWindow():
    print("Well done - you escaped \n")

# declare a subroutine called trapdoor
def trapdoor():
    print("Oh dear, you fell to your death.  Game over \n")


# declare a subroutine called startStory
# it takes no parameters and returns no value
def startStory():
    # output the story
    print(“Welcome to the first room \n”)
    print(“You are in a room, and have the choice of two doors \n

    #read the user input and store in doorChoice
    doorChoice = input(“Would you like to go through the left doo
\n")

    if doorChoice.upper() == “LEFT”:   # if the user entered left
        leftDoor()                     # call the subroutine left
    else:                              # otherwise
        rightDoor()                    # call the subroutine righ

# The main program starts here
startStory()
```

4. Read two numbers from the user; output if the first number is (or is not) g

```
# define a subroutine called method1
# it takes 2 values as parameters but does not return a value

def method1(num1, num2):
    # output the message
    print(str(num1) + " is greater than " + str(num2) + "\n")

# define a subroutine called method2
# it takes 2 values as parameters but does not return a value

def method2(num1, num2):
    # output the message
    print(str(num1) + " is not greater than " + str(num2) + "\n")

# the main program starts here
number1 = int(input("Enter a number \n"))        # read the input
number2 = int(input("Enter a second number \n")) # read the input

if number1 > number2:         # if the value in number1 is greate
    method1(number1, number2) # call subroutine method1, send num
else:                         # otherwise
    method2(number1, number2) # call subroutine method2, send num
```

5. A subroutine adds together two numbers and returns the result.

```
# declare a subroutine named calculateResult
# it takes two parameters, num1 and num2
# it returns a value

def calculateResult(num1, num2):
    # return the result of num1 + num2
    return (num1 + num2)

# the main program starts here
#call calculateResult with 10 and 20 as parameters. Store the val
result = calculateResult(10, 20)

#output the value in result
print(str(result) + "\n")
```

6. Use a subroutine to check that a value inputted by the user is valid.

```
# declare a subroutine called validateInput
# it takes one value as a parameter and returns a value

def validateInput(theNumber):
    # if the parameter value is greater than or equal to 0 and le
    if theNumber >= 0 and theNumber <= 100:
        return True   # return the Boolean True

    # otherwise
    else:
        return False  # return the Boolean False

# the main program starts here
# read the user input and store in userInput
userInput = int(input("Enter a number between 0 and 100 \n"))

if validateInput(userInput) == True:
    # if true, multiply the input by 10 and output
    userInput = userInput * 10
    print("Thanks for entering a valid number. It's now " + str(u
else:
    # otherwise output that it is invalid, and set userInput to 0
    print(str(userInput) + " is not valid. It's set to 0 instead
    userInput = 0
```

7. Three numbers are read from the user. The subroutine returns the intege multiplied by the third number, plus the second number.

```
# declare a subroutine called calculateValue
# it takes 2 values as parameters and returns a value

def calculateValue(number1, number2):
    # output the first parameter value
    print(str(number1) + " is the largest number \n")
    # read the user input and store it in thirdNum
    thirdNum = int(input("Enter third number \n"))
    #return number1 multiplied by thirdNum, add 2
    return (number1 * thirdNum) + number2

# the main program starts here

# ask the user for two numbers, store them in variables firstNum
firstNum = int(input("Enter the first number \n"))
secondNum = int(input("Enter the second number \n"))

# if the value in firstNum is greater than or equal to the value
# call calculateValue, with firstNum as the first parameter and s
# then output the value returned from the subroutine call
# else
# call calculateValue, with secondNum as the first parameter and
# then output the value returned from the subroutine call

if firstNum >= secondNum:
    print(str(calculateValue(firstNum, secondNum)) + "\n")
else:
    print(str(calculateValue(secondNum, firstNum)) + "\n")
```

8. Take as input two numbers and a calculation. Call a specific subroutine b
result of the calculation.

```
# declare a subroutine called addNumbers
# it takes two parameters as integers and returns a single number

def addNumbers(num1, num2):
    # return the result of num1 + num2
    return num1 + num2

# declare a subroutine called subtractNumbers
# it takes two parameters as integers and returns a single number

def subtractNumbers(num1, num2):
    if num1 > num2:         # if num1 is greater than num2
        return num1 - num2  # return the result of num1 - num2
    else:                   # otherwise
        return num2 - num1  # return the result of num2 - num1

# declare a subroutine called multiplyNumbers
# it takes two parameters as integers and returns a single number

def multiplyNumbers(num1, num2):
    # return the result of num1 * num2
    multiplyNumbers = num1 * num2

# declare a subroutine called divideNumbers
# it takes two parameters as integers and returns a single number

def divideNumbers(num1, num2):
    if num1 > num2:         # if num1 is greater than num2
        return num1 / num2  # return the result of num1 divided b
    else:                   # otherwise
        return num2 / num1  # return the result of num2 divided b

# the main program starts here

# read the user input and store it in firstNum
firstNum = int(input("Enter the first number \n"))

# read the user input and store it in secondNumber
secondNum = int(input("Enter the second number \n"))

# read the user input and store it in symbol
symbol = input("Would you like to +, -, / or *?  Enter the symbol

if symbol == "+":
    # if the value in symbol is a +
    # call the subroutine addNumbers, and output result
    print(addNumbers(firstNum, secondNum))
elif symbol == "-":
    # if the value in symbol is a -
    # call the subroutine subtractNumbers, and output result
    print(subtractNumbers(firstNum, secondNum))
elif symbol == "/":
    # if the value in symbol is a /
    # call the subroutine divideNumbers, and output result
    print(divideNumbers(firstNum, secondNum))
elif symbol == "*":
    # if the value in symbol is *
    # call the subroutine multiplyNumbers, and output result
    print(multiplyNumbers(firstNum, secondNum))
```

Searching

Description of Code

If data is stored in a structure such as a list, you may need to search it to find o
exists, or to find the location of an item.

There are many method of searching; some are more efficient in specific scena

A linear search goes through each item in a list, one at a time, from the first el
what it is looking for or until it reaches the end of the list. The following programs, th
implement based on these are all inspired by the linear search.

A binary search needs a list of data to be in order. It then takes the middle ele
item it is looking for. If the middle element is smaller than the item it is looki
search just on the right-hand side of the list (all the elements greater than the
it repeats it with all those elements that are smaller. This is repeated until the
finds the item it is looking for.

A range of searching methods are shown in the examples below.

Code in Context

1. Search a 1D list to find out whether an item exists, and then output if it is

```
# import the random module
import random

# declare a blank list
itemArray = []

#generate random numbers and store them in the list
for count in range(0, 10): # loop 10 times
    # generate a random number between 1 and 100 and store it in
    itemArray.append(random.randint(1, 100))

# read the user input, store in userInput
userInput = int(input("Enter the number you want to find \n"))

# set flag to be False. If false, it means the item has not been
flag = False

# loop through each element in the item and check if it is the it
for count2 in range(0, 10):
    # if the current list item is equal to userInput
    if itemArray[count2] == userInput:
        # set flag to be True
        flag = True

# if the value in flag is True, output "It was in the array"
if flag == True:
    print("It was in the array \n")
# if the value in flag is False, output "It was not in the array"
else:
    print("It was not in the array \n")
```

2. Check whether an item exists in a 1D list. If it does, output all the positio failure message if it does not exist.

```
# import the random module
import random

# declare a blank list
itemArray = []

# generate 1 random numbers and store them in the list
for count in range(0, 10): # loop 10 times
    # generate a random number between 1 and 100 and store it in
    itemArray.append(random.randint(1, 10))

# read the user input, store in userInput
userInput = int(input("Enter the number you want to find \n"))

# set flag to be False. If false, it means the item has not been
flag = False

# loop through each element in the item and check if it is the it
for count2 in range(0, 10):
    # if the current list item is equal to userInput
    if itemArray[count2] == userInput:
        # set flag to be True
        flag = True
        print("The item is at position " + str(count2) + "\n")

# if the item was not found print “It was not in the array”
if flag == False:
    print("It was not in the array \n")
```

3. Search a 1D list for an item the user inputs.

```
# import the random module
import random

# declare a blank list
itemArray = []

# generate 1 random numbers and store them in the list
for count in range(0, 10): # loop 10 times
    # generate a random number between 1 and 100 and store it in
    itemArray.append(random.randint(1, 10))

# read the user input, store in userInput
userInput = int(input("Enter the number you want to find \n"))

#loop through each element in the item and check if it is the ite
for item in itemArray:
    # if the current list item is equal to userInput
    if item == userInput:
        # set flag to be True
        flag = True

# if the item was not found print “It was not in the array”
if flag == False:
    print("It was not in the array \n")
# if it is found print “Found it”
else:
    print("Found it")
```

4. Search a list of lists for the location of an item. Output the location if fou
if it is not found.

```
# import the random module
import random

# declare a list of lists, with 10 elements by 20 elements
numbers = [[0] * 10 for i in range(19)]

# loop through the first elements
for column in range(0, 19):
    # loop through the second elements
    for row in range(0, 9):
        # generate a random number between 0 and 100 and store it
        numbers[column][row] = random.randint(0,100)

# read the user input, store in userInput
userInput = int(input("What number do you want to search for? \n")

# set flag to False. If false, means the items has not been found
flag = False

# loop through the first elements
for column in range(0, 19):
    # loop through the second elements
    for row in range(0, 9):
        # if the current item is equal to userInput
        if numbers[column][row] == userInput:
            # set flag to True
            flag = True
            # output the position
            print("The item is at position " + str(column) + " " +

# if the item was not found
if flag == False:
    # output the message
    print("It was not in the list")
```

Sort

Description of Code

If data is stored in a structure such as a list, you may need to sort it into ascendinc

There are set functions that can do this, or you can write your own sorting algorit It makes more sense to use Python’s inbuilt functions.

There are a range of sorting methods; some are more efficient than others depend

A *bubble sort* compares the first and second items in a list. If they are in the wron repeats this with the second and third items, then the third and fourth, etc. When checks whether there have been any swaps. If there have, it goes back to the first there haven’t, the list is sorted.

A *merge sort* splits each element into its own list. It then merges pairs of individu repeats this with pairs of ordered lists, and merges them into a larger, ordered list have been merged into one.

Code in Context

1. Sort the items in a list into ascending order using the sort function.

```
# import the random module
import random

items = []

# generate 1 random numbers and store them in the list
for count in range (0, 10): # loop 10 times
    # generate a random number between 1 and 100 and store it in
    items.append(random.randint(1, 100))

print(“Before \n”)

# output the contents of the list
print(items)

# sort the list into ascending order
items.sort()

print(“After \n”)

# output the contents of the list
print(items)
```

2. Sort the items in a list into descending order using the sort and reverse funct

```
# import the random module
import random

items = []
# generate 1 random numbers and store them in the list
for count in range (0, 10): # loop 10 times
    # generate a random number between 1 and 100 and store it in
    items.append(random.randint(1, 100))

print(“Before \n”)

# output the contents of the list
print(items)
# sort the list into ascending order
items.sort()

# reverse the order of the list
items.reverse()

print("After \n")

# output the contents of the list
print(items)
```

3. Sort a 2D array into ascending order by the first element in the array.

```
# import the random module
import random

# declare a list of lists, with 10 spaces by 2
items = [[0] * 2 for i in range (10)]

# generate 10 random numbers and store them in the array
# loop through the first list
for column in range (0, 9):
    # loop through the second list
    for row in range(0, 2):
        # generate a random number between 1 and 10 and store it
        items[column][row] = random.randint(1, 10)

# output the contents of the list
print("Before \n")
print(items)
print("\n")

# bubble sort the array by the first list
# true if a swap is made
swap = True

# if a swap is made in the last cycle
while swap == True:
    # set swap to be false
    swap = False
    # loop 9 times through each element in the array
    for count in range(0, 9):
        # if the current item is greater than the next
        if items[count][0] > items[count+1][0]:

            # store the current element in temporary variables
            temp1 = items[count][0]
            temp2 = items[count][1]

            # replace the current element with the next element
            items[count][0] = items[count+1][0]
            items[count][1] = items[count+1][1]

            # replace the next element with temporary variables
            items[count+1][0] = temp1
            items[count+1][1] = temp2

            # set swap to be True because a swap has been made
            swap = True

print("After \n")
# output the contents of the list
print(items)
print("\n")
```

Random Number Gene

Description of Code

You can generate a random number between any values. The random module
imported into your code. At the top of your program add the code:

```
import random
```

To generate an integer value (whole number) use the code:

```
random.randint(<lower bound>, <upper bound>)
```

Code in Context

1. Generate 100 random numbers between 1 and 100.

```
# import the random module
import random

for x in range (0, 100):         # loop 100 times
    print(random.randint(1,100)) # generate a random number betwee
```

2. Ask the user how many numbers to generate and what values to generate
random numbers, and then output each one and the total of all the gener

```
# import the random module
import random

# read the value input and store in userInput
userInput = int(input("How many numbers do you want to generate?

# read the value input and store in lowest
lowest = int(input("What is the smallest number you want generate

# read the value input and store in highest
highest = int(input("What is the largest number you want generate

# set variable total to 0
total = 0

# loop from 1 to the number the user input
for count in range (10, userInput):
    # generate a random number between the bounds entered, store
    numGenerated = random.randint(lowest, highest)
    # output the value in numGenerated
    print(str(numGenerated) + "\n")
    # add the value in numGenerated to total, store in total
    total = total + numGenerated

# output the text and the value in total
print("The numbers all added up to " + str(total) + "\n")
```

3. Generate 100 random numbers between 0 and 1 with up to two decimal p

```
# import the random module
import random

# loop 100 times
for x in range (0, 100):

    # generate a random number between 0 and 10,
    # divide the result by 10, and print to screen
    print(random.randint(0,10)/10)
```

Dictionary

Description of Code

A dictionary can be used to store multiple pieces of data with different data ty
list, but each item has an identifier that it can be referred to by.

Creating a dictionary

Once you have declared a structure, this is now a useable data type. So you ca
variable of data type – your structure name.

<dictionaryIdentifier> = {'<identifier1>': <data>, '<identifier2>': <data>, etc.}

For example:

```
horseBob = {'Name': 'Bob', 'age': 22}
```

Getting from a dictionary

You can get data from a record by using the identifier the data is associated w

<dictionaryIdentifier>['<identifier>']

For example:

```
print(horseBob['Name'])
```

Code in Context

1. Store the horse's name, year of birth, colour and height. Output the inform

```
# declare a dictionary named horse1, with four items of data -
# name, yearOfBirth, colour and height
horse1 = {'Name': '[illegible]o', 'yearOfBirth': 2012, 'colour': 'Grey', '

# output data about horse1 in a sentence
print("The " + horse1['colour'] + " horse, " + horse1['Name'] + "
str(horse1['yearOfBirth']) + " and is " + horse1['height'] + " ha
```

2. Let the user enter the favourite colour, the age and the gender for five pe
records. Each person has all three pieces of information stored in an array
age, and output the gender and favourite colour for each person.

```
# initialise a list with 5 elements
thePeople = [0,0,0,0,0]

# ask the user to enter the three pieces of information for 5 peo
for count in range (0, 5):
    colour = input("Enter person number " + str(count+1) + "'s fa
    age =int( input("Enter person number " + str(count+1) + "'s a
    gender = input("Enter person number " + str(count+1) + "'s ge
    thePeople[count] = {'colour': colour, 'age': age, 'gender': g

# calculate the total age of all list elements
totalAge = 0
for i in range (0, 5):
    totalAge = totalAge + thePeople[i]['age']

# output the average age of the 5 people
print("The average age of people entered is " + str(totalAge / 5)

# output the gender and colour for each array element
for i in range (0, 5):
    print(thePeople[i]['gender'] + " and colour " + thePeople[i][
```