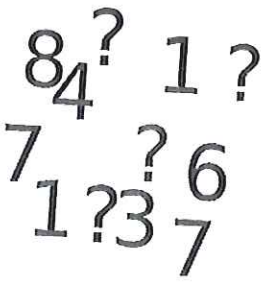
Exercise 1 – Number Game



This is a simple game whereby one person (the computer in this case) thinks of a number between 1 and 100. The other person then has to guess what the number is.

If they guess incorrectly they are given clues about whether their guess is too high or too low and they have to guess again until they get it right. The idea is to guess the number correctly in as few guesses as possible.

A program designed to play the game is shown below (and provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1     import random
2
3     def guess():
4     num = input("Please enter your guess")
5     return num
6
7     print("Welcome to the number guessing game")
8     print("The objective is to guess the number I'm thinking of.")
9     print("I will give you clues after your first guess.")
10    secretNumber = random.randint(1,100)
11    print("I have thought of a number from 1-100")
12    numGuessed=guess()
13    if numGuessed < secretNumber:
14        print("Guess is too low, guess higher!")
15    else:
16        print("Guess is too high, guess lower!")
```

Section A

A 1 The program does not run properly and you should get a syntax error related to line 4. Identify the cause of the problem and fix the program.

1 1

.....

Program updated ☐

A 2 When the program asks the user to enter their guess, it is not formatted nicely. Modify the program so that it presents a more suitable layout/prompt.

1

Program updated ☐

A 3 The welcome message does not stand out – it is merged into the request to enter their first guess and the instructions.

Modify the welcome message so that it appears underlined to separate it from the rest of the text, and then leave a blank line after the instructions before they are asked to enter their first guess.

2

Program updated ☐

A 4 Currently the program will generate a type error when you run it – this happens because the number that is being entered is actually stored as a string. On computers, you have to convert writing such as "23" to integers such as 23. This is so that the computer knows to treat it as a number. For example, "23" + "23" is actually "2323" on a computer but 23 + 23 is 46. This is why the computer needs to know whether it is a number or a string.

Find the error in your program and fix it.

1

Program updated ☐

A 5

The program only allows the user to have one guess before stopping. This is fixed using iteration.

Modify the program that it keeps asking the user to enter a guess until it is correct.

Hint: You will need to use a new variable that is initialised before the new loop.


3

Program updated ☐

A 6

The program will not perform correctly when the user guesses the number correctly. Investigate what happens and describe it, then fix the program electronically.


1 2

.....

.....

Program updated ☐

SECTION B

B 1

Develop the program further so that the game prints out the number of guesses that the user took to get the correct answer.


3

Program updated ☐

B 2

It is important to add validation to programs to prevent errors from occurring due to user input.


2

Modify the program to:

- only allow the user to enter numbers from 1–100
- print out an error message when they enter an invalid number
- ask them to try again.

Program updated ☐

B 3

Currently, if a user enters anything other than an integer, the program throws an exception.

Fix this issue in your program using a TRY...EXCEPT.


3

Remember that the guess function should not exit/return until a valid value has been entered.

Program updated ☐

Total: 20	2	18

Exercise 2 – Rock, Paper, Scissors

Rock, paper, scissors is a game played by two people in order to decide on the outcome for something, much like tossing a coin. Both players tap their fist three times and then make either a rock (clenched fist), paper (open, flat hand) or scissors (two fingers open like scissors and the others clenched). The following rules are then used to decide who wins:

- *Rock beats scissors (because it smashes them).*
- *Paper beats rock (because it wraps it).*
- *Scissors beats paper (because scissors cut paper).*
- *If both players show the same symbol then it's a draw.*

A program designed to play the game is shown below (and provided electronically).
Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1     def printRules()
2         print("The computer will think of either rock, paper or scissors.")
3         print("You will enter r for rock, p for paper or s for scissors.")
4         print("The computer will reveal it's choice and the winner.")
5        print()
6
7     def playGame():
8         choice=input("Enter r for rock, p for paper or s for scissors: ")
9         computerChoice=random.randint(0,2) # 0=rock, 1=paper, 2=scissors
10
11        if computerChoice = 0:
12            print("The computer chose: Rock")
13        elif computerChoice == 1:
14            print("The computer chose: Paper"
15        else:
16            print("The computer chose: Scissors")
17
18        if choice == r:
19            if computerChoice == 0:
20                print("It's a draw)
21            elif computerChoice == 1:
22                print("Computer Wins!")
23            else:
24                print("Player Wins!")
25
26    print("Welcome to the Rock, Paper, Scissors Game")
27    print("========================================")
28    printRules
29    playGame()
```

There are a number of syntax errors in the code, which will need to be fixed before the code can be run correctly.

A **syntax error** means that we have not followed the rules of the programming language, i.e. the command we have given is slightly wrong (e.g. a missing bracket or quotation mark).

Section A

A 1 The first error is on line 1 which defines the function printRules().
Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 2 There is a second syntax error within the printRules() function.
Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 3 In Python, we use = to assign a value to a variable but we use == to compare the value of one variable to another (or against a specified value).

Find a place where an incorrect number of = has been used and fix the program accordingly. State the line number below.

1 1

.....

Program updated ☐

A 4 There is another syntax error on line 14. Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 5 There is a problem on line 18. Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 6 There is another syntax error on line 20. Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 7 There is one final syntax error on line 28. Identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 8 The program is now giving a name error, random not defined.
Describe the issue below and fix the program accordingly.

1 1

.....

Program updated ☐

A 9 There is now a logic error in the program. Test it by playing the game and choosing all the options to see what happens. Identify the issue and fix the program accordingly.

1 2

.....

Program updated ☐

SECTION B

B 1 The player can currently enter something other than r, p or s. If they do so, the program should be robust and ask them to re-enter their choice. Develop the program further by implementing this validation rule.

2

Program updated ☐

B 2 The user should be allowed to enter an uppercase R, P or S, not just a lowercase one. Develop the program further to implement this additional functionality.

2

Program updated ☐

B 3 The program currently only plays the game once. Modify the program so that it asks the user if they would like to play again instead of just pressing enter to exit.

2

Program updated ☐

Total: 25 9 16

Exercise 3 – Turtle Drawing

One of the first robots invented for drawing was the turtle – a simple robot with a choice of pens that drives around the floor and either has a pen touching the paper underneath it (using different colours), or has the pen raised so that it can move to a new location to start the next part of the drawing (or the next drawing).

To create turtle drawings, Python's turtle package is used. The package is called 'turtle', as it allows you to control a 'virtual turtle' that draws lines where the turtle has moved to.

*This is an example of an **external library**. Some basic commands have been given below.*

Command	Description	Example
`turtle.forward(x)`	Move x pixels in the direction the turtle is pointing.	`turtle.forward(5)`
`turtle.left(x)`	Turns the turtle left x degrees.	`turtle.left(90)`
`turtle.right(x)`	Turns the turtle right x degrees.	`turtle.right(90)`
`turtle.color(x)`	Sets the colour using a value represented using hexadecimal.	`turtle.color('#f8008f')`
`Turtle.penup()`	Puts the pen up which means that the turtle can be moved without creating a line.	`turtle.penup()`
`Turtle.pendown()`	Puts the pen down which means that the turtle will draw a line when it moves.	`turtle.pendown()`
`turtle.heading()`	Returns the direction in which the turtle is heading (in degrees). If the turtle is heading east, it's heading is 0.	`turtle.heading()`
`turtle.setheading(x)`	Sets the direction in which the turtle is facing.	`turtle.setheading(90)`
`turtle.speed(x)`	Allows the speed of the turtle to be changed (1 = slow, 10 = fast).	`turtle.speed(10)`

A basic program that draws a square has been given below.

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1     import turtle
2
3     BLUE="#0000ff"
4     PINK="#ff00ff"
5     GREEN="#00ff00"
6
7     def drawSquare(size, colour):
8         turtle.color(colour)
9         for i in range(4):
10            turtle.forward(size)
11            turtle.right(90)
12
13    turtle.speed(5)
14    turtle.setheading(0)
15    turtle.pendown()
16    drawSquare(100, BLUE)
17    turtle.penup()
18    turtle.forward(5)
19    turtle.right(90)
20    turtle.forward(5)
21    drawSquare(80, PINK)
22
23    turtle.exitonclick()
```


SECTION A

A 1 The program draws a blue square and then the turtle attempts to draw a pink square, but no square appears. Identify the issue and fix the program so that the pink square is drawn.

1 1

.....

Program updated ☐

A 2 After the bug described above has been fixed, you notice that the pink square is drawn in the wrong place – it should be inside the blue square.

By looking at the direction of the turtle, identify the issue and fix the program accordingly.

1 1

.....

Program updated ☐

A 3 Modify the program to make the colour of the inside square green instead of pink. You should only change one line.

1

Program updated ☐

A 4 The colour codes on lines 3–5 are in hexadecimal. Modify the program by adding a new constant storing the hex colour for RED (look it up if you need to), then change the colour of the outside square to RED.

2

Program updated ☐

A 5 The position of the inside square is not even. Modify the program so that the inside square is positioned evenly inside the outside square.

1

Program updated ☐

SECTION B

B 1 Programmers cannot be expected to know how every function of every library works, but they need to be able to look up the documentation to fill any gaps in knowledge.

1 1

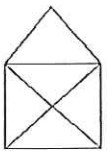
By looking up the documentation for the turtle module, modify the program by adding a command that will make turtle invisible after the image has been drawn. Write down the command used below.

https://docs.python.org/3.5/library/turtle.html

.....

Program updated ☐

B 2 Other shapes can be drawn easily using the turtle. What isn't quite so easy is drawing a shape without taking the pen off the paper, or drawing the same line twice. One example that can be achieved is shown on the right.



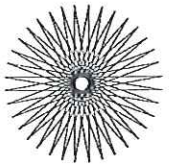
3

Create a function called drawHouse() to draw this shape using turtle.

You can start at any point you like. You will need to use Pythagoras' theorem to calculate the size of the lines to draw – a length of 100 is recommended for the main house.

Program updated ☐

B 3 Create another draw function to replicate the star shown on the right. Use a loop to create the shape. Attempt to colour in the star using the fill commands.



3

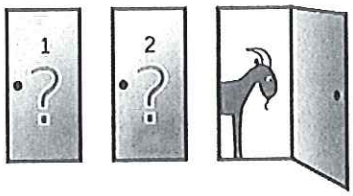
Program updated ☐

Total: 16 3 13

Exercise 4 – The Monty Hall Problem

Consider the following scenario:

You are on a TV game show, and have the choice of three doors to open. One of the doors has a brand-new car behind it – the other two have old goats behind them.



Once you have picked a door, the game show host opens one of the two doors that you did not pick, to show you a goat. He then offers you the choice to switch your choice to the remaining door.

Should you switch your choice? Does it make any difference to how likely you are to win the car? Remarkably, if you do not switch you have a chance of 1/3, and if you do switch, the odds double and the chance you win is 2/3!

You will be running a simulation of the Monty Hall Problem. A basic program is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1     door = ["goat","goat","car"]
2
3     choice = input("Door 1, 2 or 3? ")
4     otherDoor = 0
5     goatDoor = 0
6
7     if choice == 1:
8         if door[1] == "goat":
9             otherDoor = 3
10            goatDoor = 2
11        elif door[2] == "goat":
12            otherDoor = 2
13            goatDoor = 3
14    elif choice == 2:
15        if door[0] == "goat":
16            otherDoor = 3
17            goatDoor = 1
18        elif door[2] == "goat":
19            otherDoor = 1
20            goatDoor = 3
21    elif choice == 3:
22        if door[0] == "goat":
23            otherDoor = 2
24            goatDoor = 1
25        elif door[1] == "goat":
26            otherDoor = 1
27            goatDoor = 2
28
29    switch = input("There is a goat behind door " + goatDoor + \
30                   " switch to door " + otherDoor + "? (y/n) ")
31
32    if switch == "y":
33        choice = otherDoor
34
35    if door[choice-1] == "car":
36        print("You won a car!")
37    else:
38        print("You won a goat!")
```

Section A

A	1

Describe the purpose of the '\' symbol used on line 29.

1

.....

A 2

After making a choice, the program crashes unexpectedly.

Describe the reason for this, and fix the program accordingly.


1 1

.....

.....

Program updated ☐

A 3

The program does not recognise a user's choice, even if it is valid.

Describe the reason for this, and fix the program accordingly.


1 1

.....

.....

Program updated ☐

A 4

Currently, the prize is always behind the same door.

Fix this electronically by using random.shuffle().


1

Program updated ☐

A 5

On line 35, the selection statement refers to **`choice-1`**.

Why does it refer to this instead of simply **`choice`**?


1

.....

SECTION B

These tasks are to enable you to build a simulation or computer model for the Monty Hall problem. A computer model will run automatically without the need for you to intervene in each situation. It will also mean that you can choose and set different variables, such as how many times to play the game or whether you wish to switch doors or not. The idea is that you can then automatically run the model for thousands of games and see what happens.

B 1

To observe the effect of switching doors, it would be useful to be able to play multiple games. Modify the program so that 10 games are played before the program ends.

Name the type of programming construct used to achieve this below.


1 1

.....

Program updated ☐

B 2

In a simulation, you should not need to manually enter your choice. Instead, we use randomness to pick for us.

Change the program so that instead of asking for an input, it automatically chooses one at random.


1

Program updated ☐

B 3

Create a subroutine montyHall() using the code that you have written so far. The subroutine should take in two parameters – the first should indicate the number of games that should be played, and the second should indicate whether you should switch doors for those games or not.


2

Run it 1,000 times with switching, and 1,000 times without switching. In each test you should output the number of times that you won the car, and how many times you won a goat.

Program updated ☐

Total: 12 5 7

Exercise 5 – Caesar Cipher

Life in Ancient Rome was very different to how we live in the present day. In 44 BC, there were no computers, no cars, and if you wanted to send a message to your friend who lived in a different city, you either had to deliver it yourself or pay someone to deliver it for you.

*The problem that Julius Caesar had, during the sending of messages, was that military secrets were being stolen or read during delivery. To combat this issue, he devised one of the very first examples of **encryption**. In this method, each letter is shifted along by a fixed amount to turn the plaintext into ciphertext. If the secret key is 2, then A → C, B → D, ..., Y → A and Z → B.*

Encryption: The act of scrambling a message in a way that only the intended recipient can read it.

A program designed to perform a basic Caesar cipher is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1       def letter_to_number(letter):
2        letters = "abcdefghjiklmnopqrstuvwxyz"
3        # Find the number corresponding to the given letter.
4        # In this case a = 0, b = 1, c = 2, ..., z = 25.
5        number = letters.index(letter)
6        return number
7
8       def number_to_letter(index):
9        letters = "abcdefghijklmnopqrstuvwxyz
10       # Finds the letter corresponding to the given number.
11       # In this case 0 = a, 1 = b, 2 = c, ..., 25 = z.
12       return letters[index]
13
14      def shift(letter):
15       n = letter_to_number(letter)
16       return number_to_letter( (n + 13) % 26 )
17
18      def rot13(string):
19       ciphertext = ""
20       for letter in string:
21       ciphertext += shift(letter)
22       return ciphertext
23
24
25      plaintext = "i love computing!"
26
27      ciphertext = rot13(plaintext)
28      print(ciphertext)
```

Section A

A 1 What is the value that each letter has been shifted by in this program?

1

.....

A 2 Initially, the program does not run correctly.

Identify the cause of the error (including naming the type of the error) and fix the program accordingly.

2 1

.....

Program updated ☐

A 3 A table has been provided that will let you manually decode the given ciphertexts.

0	1	2	3	4	5	6	7	8	9	10	11	12
a	b	c	d	e	f	g	h	i	j	k	l	m
13	14	15	16	17	18	19	20	21	22	23	24	25
n	o	p	q	r	s	t	u	v	w	x	y	z

The output from the program is '**w ybir pbzchgwat!**'.

By decoding the message using the table above, you can deduce that there is an error in the program. Fix the error electronically, and write the line that it occurred on below.

.....

Program updated ☐

A 4 To encrypt another message, you have to change the program's source code. Ideally, the program should ask the user to input a string to be encrypted instead.

State below how you would ask the user for input in Python. Update your program electronically to reflect this.

.....

Program updated ☐

A 5 When a string is entered that contains uppercase letters, the program will not run correctly. Change your program electronically to allow the user to input uppercase characters without the program failing. State the function that you used below.

.....

Program updated ☐

SECTION B

B 1 ROT13 is a special case of the Caesar cipher, as to decrypt a previously encrypted message you simply run the function rot13() on it again. If the shifting value is different from the one used in ROT13, to decrypt a message you need to shift the letters in the other direction (a negative amount).

Create a new function, encrypt(), that takes a string and a 'shift value' as parameters, that shifts the letters by the given shift value.

You will need to change the function shift(). You will also need to change your rot13() function.

Below, write the encryption of the phrase 'hello world!' with a 'shift value' of 2.

.....

Program updated ☐

TRIVIA (Attention all Mathematicians!!)

The Python command % is often referred to in pseudo code as MOD. MOD returns the remainder after integer division has occurred. The function `10 MOD 5` will return 0 but `11 MOD 5` will return 1.

Although the function `-9%3` returns 0 as expected, `-10%3` returns 2 whereas `10%3` returns 1.
The function `-5%20` does not return −5 or 5, as you might expect. What does `-5%20` return?

Why do you think that is (mathematically)?

Hint: Computers cannot do division (or multiplication) – they use repeated addition.

B 2


1 3

To decrypt a message, you need to shift the letters in the opposite direction. Write a function decrypt() that takes in a string and a 'shift value', that shifts the letters in the opposite direction.

Using your function, decrypt the string "drsc sc k combod wocckqo!" that has been encrypted by shifting the letters 10 spaces forwards.

.....

Program updated ☐

B 3


3

If a user needs to encrypt a lot of text, it is more sensible to read it from a text file. Changing lines 25–27, load the text file "plaintext.txt" into your program and encrypt it using a "shift value" of 13.

Program updated ☐

Total: 22	8	14

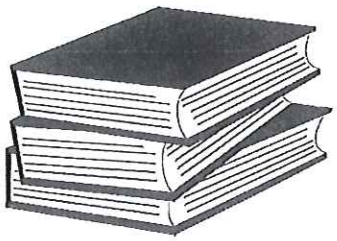
Exercise 6 – Check Digits

At many stages of communication between hardware devices there is a chance that the message will be partially misunderstood. Check digits are especially useful, as they can tell us if there has been an error in transmission.

One use of check digits is on ISBN numbers, used for book identification. Each new book is given a unique 12-digit code. The thirteenth digit is reserved for a check digit that verifies that the ISBN is valid.

To calculate the check digit of a 12-digit unique book identifier, the following arithmetic operations are performed:

Split the unique identifier:	9 7 8 1 4 7 1 1 1 7 9 0
Multiply every other number by 3:	9 21 8 3 4 21 1 3 1 21 9 0
Add all of the numbers:	9+21+8+3+4+21+1+3+1+21+9+0 = 101
Perform division modulo 10:	101 MOD 10 = 1
Subtract this result from 10:	10 – 1 = 9 ← This is the check digit.



NOTE: if the sum modulo 10 = 0, the check digit is 0.

This particular ISBN is written in the following way: 978-1-47-111790-9

The first number, 978, indicates that the 13-digit code is used for identifying a book. There are other identifiers used for different document types; for instance, the number 979 is used for sheet music, which uses the same check digit system.

The second number, 1 (after the dash), indicates the book is from an 'English-speaking area'. There are different codes depending on the country – the first few have been listed below.

0	English-speaking area	2	French-speaking area	4	Japan
1	English-speaking area	3	German-speaking area	5	(former) USSR

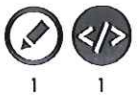
The third and fourth numbers identify the publisher, the book, and the edition of the book. Finally, the last number is the check digit used for error checking. *An ISBN checking program has been given below (and electronically).*

```
1     def ISBNcheck(ISBN):
2
3         # Split the ISBN into the Unique ID and the check digit
4         unique_id = []
5         for i in range(len(ISBN)-1):
6             unique_id.append(int(ISBN[i]))
7         actual_check_digit = ISBN[-1:]
8
9         # Multiply the second, fourth, sixth, ... elements by three
10        times_three=[]
11        for x in range(len(unique_id)):
12            if x%2 == 0:
13                times_three.append(unique_id[x]*3)
14            else:
15                times_three.append(unique_id[x])
16
17        # Calculate the sum of the numbers
18        sum_new_digits = 0
19        for x in range(0,len(times_three)):
20            sum_new_digits += times_three[x]
21
22        # Take the sum mod 10, and subtract from 10
23        sum_mod_10 = sum_new_digits % 10
24        if sum_mod_10 == 0:
25            sum_mod_10 = 10
26        check_digit = 10 - sum_mod_10
27
28        # Check that the calculated check digit is equal to the given check digit
29        if check_digit == actual_check_digit:
30            return "valid."
31        else:
32            return "invalid."
33
34    choice = input("Enter the ISBN number: ")
35    print("ISBNcheck() returns " + ISBNcheck(choice))
```


SECTION A

A 1

The code on line 7 takes the last character of the array and assigns it to the check digit. There is a problem with the data types on this line which will mean that every ISBN will return invalid.



Fix the error electronically, and write the line that it occurred on below.

.....

Program updated ☐

A 2

Testing your program on the (valid) ISBN 9781471117909, you'll see that the program does not work correctly. The error is in the iteraction statement that starts on line 12.

Explain below why the error is happening, and fix your program electronically.

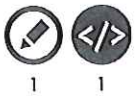
.....

.....

Program updated ☐

A 3

Finding the value of sum_new_digits on lines 11–13 can actually be written in one line instead of three, using an inbuilt function.



Change lines 11–13 so the sum of the array is calculated in one line.
State below the name of the inbuilt function that you have used.

.....

Program updated ☐

A 4

In this program the user can enter a 13-digit number and check that the check digit (the last digit) is correct. Add length validation to your program to only continue if the user has entered a 13-digit number.

Program updated ☐

A 5

The program currently exits immediately before you can see the output.

Describe how this could be prevented, and implement this change to the program.

.....

.....

Program updated ☐

SECTION B

B 1

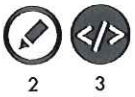
Instead of entering a 13-digit number, the user should enter a proper ISBN string in the format 978-x-x-x-x, where, excluding the dashes, there are still thirteen digits. Change your program to reflect this – a message should show if the ISBN is invalid.

Hint: The functions .split() and "".join() will be useful.

Program updated ☐

B	2

Explain why returning "True" and "False" is better than returning "valid." and "invalid." Change your program electronically to do this.



2 3

.....

.....

Program updated ☐

B	3

The second number in the ISBN (after the first dash) denotes either the language of the book, or the country from which the publisher originates.

4

If the ISBN is valid, your program should print out the country that the ISBN belongs to.

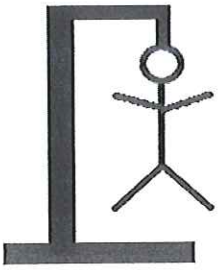
To do this, write a function called getCountry(), that converts the second number into a country. The first six codes have been given below (you can assume that any given ISBN is from one of these countries).

0	**English**
1	**English**
2	**French**
3	**German**
4	**Japan**
5	**(former) USSR**

Program updated ☐

Total: 24 | 7 | 17

Exercise 7 – Hangman



Hangman is a popular pencil and paper game where one player has to guess the word set by the other, one letter at a time. If the guessing player guesses incorrectly, a stick figure being hanged is drawn one line at a time. The game is over after a certain number of guesses.

The origins of Hangman are fairly unknown, with a guess that it was first played during the Victorian era, as hanging was one of the more popular methods of execution at the time. Why a game needed to be created about it is a mystery, however!

*To play this game on a computer, we store the words as **strings**.*

String: A data type that can be seen as a list of characters.

A program that performs functions needed to play hangman is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1   print("Welcome to Hangman!\n")
2
3   word = list("computing")
4   guessed_word = list("_________")
5   lives = 10
6   wordGuessed = False
7
8   while lives>=1 and not wordGuessed:
9
10      print(" ".join(guessed_word))
11
12      user_guess = input( "Guess a letter/word! (" + str(lives) + \
13                          " lives remaining)\n")
14
15      # Check if letter is in the word
16      letter_in_word = False
17      for i in range(len(word)):
18          if user_guess = word[i]:
19              guessed_word[i] = user_guess
20              letter_in_word = True
21
22      if letter_in_word == False:
23          lives -= 1
24
25      if guessed_word = word:
26          print(" ".join(guessed_word))
27          print("You have guessed the word correctly!")
28          wordGuessed=True
29      elif lives > 1:
30          print("You failed to guess the word correctly :(")
```

Section A

A 1 Initially, when the program is run, an error immediately occurs. Explain the error below, and fix it electronically.

1 1

.....

.....

Program updated ☐

A 2 Explain the purpose of the characters "\n" and "\" in lines 12–13.

2

.....

.....

A 3

The program tells you that you have lost after your first guess, this is due to a logic error. Fix your program electronically and state the line(s) that you have changed below.

1 1

.....

.....

Program updated ☐

A 4

Describe why a guess of 'C' does not do anything, even though it is in the string "computing". Fix your program on the computer.

1 1

.....

.....

Program updated ☐

A 5

Describe the purpose of the join() function on line 10 of the program.

2

.....

.....

SECTION B

B 1

The user should also be able to guess the whole word, as well as the individual letters. Remarkably, this added functionality only requires changing one line of code.

2

By changing **one line** of your program, add functionality that compares the user's guess to the word – if they match, the game should stop.

Note: There is already code that does something similar to this – what can you add to this code to include this extra requirement?

Program updated ☐

B 2

So far, the Hangman game only uses one word – "COMPUTING". This makes the game pretty boring – it would be much better if multiple words could be used.

7

In your electronic program, change lines 3 and 4 to read a random word from the provided text file, "words.txt".

Hint: You will also need to change the way that the underscore mask is created.

Program updated ☐

B 3

Finally, it would be useful if the user could see a list of all of the previous letters that they have entered.

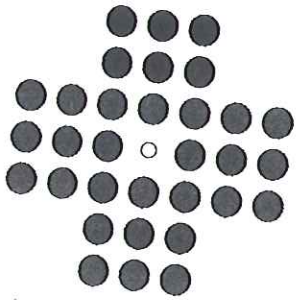
4

Add an array called 'guessed_letters' to your program, that stores the user's guess if it is **unsuccessful**. You will also need to print out this list for the user after each guess.

Program updated ☐

Total: 23 7 16

Exercise 8 – Peg Solitaire



Peg solitaire is a one-player game consisting of a board with 33 holes and 32 pegs. The aim is to remove pegs by jumping over them, in a similar fashion to draughts (or checkers).

For example, if in a line you had **abXc** *– with X representing a space – you could move the leftmost peg into the space, removing the 'b' peg (making* **XXac***).*

You will be making an electronic version of the game, but on a 4 × 4 board.

A basic Python peg solitaire game is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

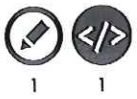
```
1   def find(board,choice):
2       for i in range(1, len(board)):
3           for j in range(1, len(board[0])):
4               if choice == board[i][j]:
5                   return i, j
6
7   grid = [["a","X","c","d"],["e","f","g","h"],["i","j","k","l"],["m","n","o","p"]]
8
9   while True:
10      print(grid)
11      choice = input("Enter a letter: ")
12      direction = input("Enter a direction (u,d,l,r): ")
13
14      row, column = find(grid, choice)
15
16      if direction == "r":
17          if column + 2 < len(grid[0]):
18              if grid[row][column + 2] == "X":
19                  grid[row][column] = "X"
20                  grid[row][column + 1] = "X"
21                  grid[row][column + 2] = choice
22
23      if direction == "l":
24          if column - 2 >= 0:
25              if grid[row][column - 2] == "X":
26                  grid[row][column] = "X"
27                  grid[row][column - 1] = "X"
28                  grid[row][column - 2] = choice
29
30      if direction == "u":
31          if row - 2 >= 0:
32              if grid[row - 2][column] == "X":
33                  grid[row][column] = "X"
34                  grid[row - 1][column] = "X"
35                  grid[row - 2][column] = choice
36
37      if direction == "d":
38          if row + 2 < len(grid):
39              if grid[row + 2][column] == "X":
40                  grid[row][column] = "X"
41                  grid[row + 1][column] = "X"
42                  grid[row + 2][column] = choice
```

NOTE: *In this program, it will not close itself unless you have completed the tasks in Section B. To manually exit the program press Ctrl+C.*

SECTION A

A 1

The 'find' function cannot find letters on the first column or the first row.
Explain below why this is the case. You should also fix the problem electronically.


1 1

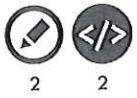
.....

.....

Program updated ☐

A 2

Instead of showing 'X' for the blank spaces, it has been decided that a blank space should be used instead.


2 2

Describe why the current program is not particularly good if you need to change the space symbol frequently, and how the program could be improved.

Change your program appropriately, making the consideration that it might be changed again in the future.

.....

.....

Program updated ☐

A 3

The program does not print the 2D array in a readable manner.

Write a function show() that takes the board as a parameter, and prints it in a grid.

You should change line 10 so your function is actually used.


4

Program updated ☐

A 4

Add validation to your program to ensure that a choice is in the correct format/case.
(For instance, entering "C" and "R" on the first turn does not move 'c' left.)


1

Program updated ☐

A 5

Having an IF statement inside an IF statement inside an IF statement is not particularly good programming practice.

State below the proper name that we give to 'IF inside an IF inside an IF' statements, and change your program so this practice is not used.


1 2

.....

Program updated ☐

SECTION B

B 1

When moving a peg, the program assumes that the space next to it is also a peg.

Explain why this happens, and fix your program to only make a move if the chosen peg is jumping over a different one.


1 4

.....

.....

Program updated ☐

B	2

It would be useful if the player could save their progress in the game. If the user inputs the word 'save' when asked for a letter, the program should save the state of the current game.

11

You should save the board by storing the size and then the contents of the board. Write the number of rows onto the first line of the file and the number of columns onto the second line. Then write the contents of each space on a new line in a text file called "game.txt".

You should also allow the user to load the game using the by inputting the word 'load' – this should read the number of rows and columns and set the board size appropriately, and then read in the contents of the board.

Program updated ☐

B	3

The board can be extended to include larger sizes. Currently, the grid is hard-coded into the program. Instead, it should be generated at the start of the program.

3

Given that the ord('a') = 97, extend your program to allow custom grid sizes of up to 6 × 4. You should set the space that would have the letter 'b' to be the empty space in any case.

The grid size should be set in the program – the user does not need to input the grid dimensions.

Program updated ☐

B	4

The game has been won if there is one piece remaining.

3

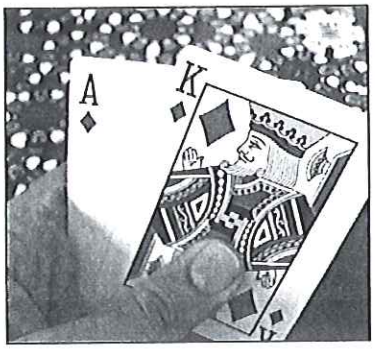
Add a check to your program that tests if the game has been won.

Test that your function works by playing on a 4 × 1 grid.

Program updated ☐

Total: 36	5	31

Exercise 9 – Blackjack Hands



Blackjack is one of the more well-known card games, played all over the world by millions of people in their homes and in casinos. It is a fairly unique game, in that the dealer only has a slight edge over the player – and players can actually gain a slight advantage if they learn how to count cards (although this is frowned upon by gambling establishments!).

The given program automatically plays blackjack. It should repeatedly 'hit' (get a new card) and only 'stick' (stop hitting) when the sum of the cards is higher than 17 (J, Q and K are each worth 10 points, A is worth 1 or 11 points). The aim is to get as close to 21 as possible; however, if you go over you are 'bust' (you have lost).

A basic blackjack program is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
import random

def getValue(card):
    try:
        return int(card)
    except:
        if card == "J" or "Q" or "K":
            return 10
        else:
            return 11

print("Automatic Blackjack Player\n")

games = 0
gameOver=False

while not gameOver:

    deck = ["A", "2", "3", "4", "5", "6", "7", "8", "9", "10", "J", "Q", "K"] * 4
    random.shuffle(deck)

    hand = []
    score = 0

    while score < 17 and len(deck) != 0:
        card = deck.pop()
        hand.append(card)
        score = score + getValue(card)

    if score == 21:
        print("Blackjack!")
        games += 1
    if score < 21:
        print("You have scored " + str(score))
        games += 1
    else:
        print("Uh oh, you have gone bust!")
        games += 1

    print("Your cards were " + hand + "\n")

    if len(deck) < 1:
        gameOver=True

number_of_games = games
print("\nYou played " + str(number_of_games) + " games.")
```


SECTION A

A 1

Initially, the program can correctly add the values of a hand, but crashes when it tries to print the hand.

Explain why, and fix the issue electronically.

.....

.....

Program updated ☐

A 2

Currently the program gets stuck in an infinite loop.

Explain why, and stop your electronic program from getting stuck in the loop.

.....

.....

Program updated ☐

A 3

Explain the purpose of the code `* 4` when creating the deck variable on line 19; why is it good practice to create the deck in this way?

.....

.....

.....

A 4

There is currently a logic error in the getValue() function – all picture cards are given the value 10. Identify below where the error occurs in this function, and fix it electronically.

.....

.....

Program updated ☐

A 5

If you hit Blackjack, a message appears stating that you have gone bust.
State the line number where the error occurs, and fix your program electronically.

.....

Program updated ☐

SECTION B

B 1

In addition to the number of games played, it would be useful to know how many times you hit blackjack, how many times you scored higher than 17 (and less than 21), and how many times you went bust.

Change the variable games to a list. It should store [number of blackjacks, number of games (> 17), number of busts] in that order.

The program should print out the percentage of the times that each outcome occurs.

Program updated ☐

B 2

In blackjack, the ace can either have the value 11, or the value 1. At the moment the program only sees it as the value 11.

Change your program so that instead of going over 21 by counting an ace as 11, it counts it as 1 instead, before continuing as normal.

Program updated ☐

3

B 3

In order to see how close the game is in terms of the edge that the dealer has, you need to run the game over a much longer period of time and get the computer to generate a hand too. This is called modelling.

Change the number of decks to 500 – don't print out each hand, just the percentages of hands won with blackjacks, hands lost with blackjacks, hands won with a total >= 17, hands lost with a total >= 17 and total busts.

Program updated ☐

10

Total: 27	7	20

Exercise 10 - Connect Four

Connect Four is a two player logic game in which players take it in turns to drop a coloured piece of plastic into a grid, until one player has four colours matched in a row, column or diagonal.

*To represent this grid on a computer, we can use **two-dimensional arrays**. Then, each value in the array can either be an 'R' or a 'B', depending on whether the tile in that space is red or black.*

Array: A data type that can hold multiple values of the same data type.

A program that performs functions needed to play the game is shown below (and is provided electronically).

Study the code and try to understand what is happening in the program, before attempting the questions that follow.

```
1   def draw(grid):
2       print("")
3       print("1 2 3 4 5") # Print column headers
4       print("| | | | |")
5       print(grid[0][0], grid[0][1], grid[0][2], grid[0][3], grid[0][4], "- row 1")
6       print(grid[1][0], grid[1][1], grid[1][2], grid[1][3], grid[1][4], "- row 2")
7       print(grid[2][0], grid[2][1], grid[2][2], grid[2][3], grid[2][4], "- row 3")
8       print(grid[3][0], grid[3][1], grid[3][2], grid[3][3], grid[3][4], "- row 4\n")
9
10  def add_piece(grid, column, row, player):
11      if player == 1:
12          piece = "B"
13      else:
14          piece = "R"
15      grid[row][column] = piece
16      return grid
17
18  #- MAIN PROGRAM -------------------------------------------
19
20  board = [[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0],[0,0,0,0,0]]
21  won = False
22
23  draw(board)
24
25  while not won == True:
26
27      player = 1
28
29      print("It is player " + str(player) + "'s go.")
30      c_choice = int(input("Enter the column number. "))
31      r_choice = int(input("Enter the row number. "))
32
33      board = add_piece(board, c_choice, r_choice, player)
34
35      if player == 1:
36          player = 2
37      else:
38          player = 1
39
40      draw(board)
41
42  print("Player " + str(player) + " has won!")
```

***NOTE:** In this program, it will not close itself unless you have completed the tasks in Section B. To manually exit the program press Ctrl+C.*

SECTION A

A 1

The first error that you might notice is that the player does not change every turn. Describe below why this happens, and then fix your program electronically.

.....

.....

Program updated ☐

A 2

The column and row selection also does not work correctly. Describe why this does not work below, and fix the problem on your electronic copy of the program.

.....

.....

Program updated ☐

A 3

Try entering a negative column, or a column greater than 5. Describe below what happens in the program when you try to enter a negative column.

.....

.....

Add validation to the program after lines 30 and 31 to ensure that the choice of column or row is valid. The program should repeatedly prompt the player to enter a number until it is valid.

Program updated ☐

A 4

You can currently overwrite other people's moves by choosing the same space as them. Describe below the action you should take to stop this from happening, and update your electronic program to reflect this. *If the space is not empty, a message should be shown saying that the player has forfeited their turn.*

.....

.....

Program updated ☐

A 5

In the real game, you cannot put the game pieces anywhere – they either need to be on the bottom row, or on top of another existing piece.

Instead of asking for a row number, it makes more sense to only ask for a column, and then place the piece in the lowest empty tile on the board. Change your program so that it no longer requires the user to enter a row, and places a piece in the lowest free space on that column.

By modifying the add_piece() function, add verification to only allow a move to be made if the column has at least one empty space. If a move is invalid, you should print a suitable message and let the next player take their turn.

Program updated ☐

SECTION B

B 1

The grid in Connect Four actually uses seven columns and six rows.


3

Change the draw() function to allow *any* sized grid to be drawn. *The size of the grid should be calculated from the grid parameter.*

Test that it works by changing line 20 to:

```
board = [[0,0,0,0],[0,0,0,0],[0,0,0,0],[0,0,0,0]]
```

You may assume that the number of rows or columns will not be greater than 9.

Program updated ☐

B 2

Change the program so that it asks the user to enter the size of the board (maximum 9 × 9) and then initialises the board correctly.


4

Program updated ☐

B 3

Your program currently runs forever. Describe the reason for this.


1 8

.....

.....

To fix this, some checking needs to be put in place to determine whether someone has won the game. Write a function check_winner() that checks whether there are four pieces in a row, *horizontally* or *vertically*.

Use this function directly after adding a piece to stop the program if there are four pieces in a row (horizontally or vertically).

For two additional marks, extend this to check for upward diagonal winning lines and, for a final two marks, extend this to check for downward diagonal winning lines.

Program updated ☐

Total: 29 | 5 | 24

