

<https://editor.raspberrypi.org/en/projects/blank-python-starter?authuser=0>

My git repo with all the code:

<https://git.uthmn.com/boiabba/Programming-Challenges>

EXERCISE 01: CALCULATE USING PYTHON

Use the Python interpreter 'shell' and the `print()` function to complete these exercises:

- | | |
|-----------------|------------|
| 1. $50-5*6$ | 6. $21\%6$ |
| 2. $(50-5*6)/4$ | 7. $17//5$ |
| 3. $8/5$ | 8. $5*3+2$ |
| 4. $4*3$ | 9. $5**2$ |
| 5. $12/3$ | 10. $1//2$ |

main.py	Text output
1 <code>print(50-5*6)</code>	20
2 <code>print((50-5*6)/4)</code>	5.0
3 <code>print(8/5)</code>	1.6
4 <code>print(4*3)</code>	12
5 <code>print(12/3)</code>	4.0
6 <code>print(21&6)</code>	4
7 <code>print(17//5)</code>	3
8 <code>print(5*3+2)</code>	17
9 <code>print(5**2)</code>	25
10 <code>print(1//2)</code>	0

EXERCISE 02: VARIABLES AND ASSIGNMENT

1. Which of these variable names and assignment statements will NOT cause a syntax error?

variable names and assignment statements

```
continue_list = 15          #(a)
red Counter = True         #(b)
154.2 = length             #(c)
_1234 = 'password'         #(d)
my_best_friend = "Thomas"  #(e)
```

2. Write a program to calculate the area of a circle with a radius of 6.7 cm and print the result of the calculation.
 - a. Set the value of $\pi = 3.142$
 - b. The area of a circle is $\pi \times \text{radius squared}$
3. Write a program to store the following variables:
 - a. Forename
 - b. Surname
 - c. Age
 - d. Height

Use one print function to display the information in a sentence.

1. `continue_list = 15`

```
from decimal import Decimal
import math

# Use Decimal for bcd (binary coded decimal) to avoid floating point errors
radius = Decimal('6.7')

pi = Decimal(str(math.pi))

area = pi * (radius ** 2)

print(f"{area}cm2")
```

- 2.

```
141.02609421964580777cm2
```

```
name = input("Please enter your name: ").split(" ")
forename = name[0]
surname = name[-1] # handle the case where a middle name is supplied

age = int(input("How old are you? "))

height = int(input("How tall are you (cm)? "))

def display_info(forename, surname, age, height):
    print(f"Hello {forename.capitalize()} {surname.capitalize()}! You are {age} years old and {height}cm tall.")
    # uthman fatih to ufatih
    print(f"Your username is {forename.lower()[0]}{surname.lower()}")
```

3. `display_info(forename, surname, age, height)`

```
Please enter your name: John Richard Doe
How old are you? 23
How tall are you (cm)? 172
Hello John Doe! You are 23 years old and 172cm tall.
Your username is jdoe
```

EXERCISE 03: STRINGS

Complete these exercises in script mode in IDLE.

1. Prompt the user to enter their name. Output a message: 'Hello (name), I hope you are well'; the message should be output on two lines using one print statement.
2. Prompt the user to enter two whole numbers. Output a message 'The average of your two numbers, (number 1) and (number 2) is (average)'.
3. Prompt the user to enter a string for encryption and a key value between 1 and 26. Output the original string and the encrypted string with a suitable message.

```
name = input("Please enter your name: ").split(" ")
for i in range(len(name)):
    name[i] = name[i].capitalize()
name = " ".join(name)
print(f"Hello {name}! \nI hope you are doing well.")
```

1.

```
[kingvcheese@cachyos-x8664 school]$ python3
Please enter your name: UthMan fatih
Hello Uthman Fatih!
I hope you are doing well.
```

```
numbers = input("Please enter two numbers: ")

# if numbers variable containers , split by , else split by space
if "," in numbers:
    numbers = numbers.split(",")
else:
    numbers = numbers.split(" ")

# exit if len(numbers) != 2
if len(numbers) != 2:
    print("Please enter two numbers")
    exit()

# convert to int
for i in range(len(numbers)):
    numbers[i] = int(numbers[i])

# calculate the average
average = sum(numbers) / len(numbers)

print(f"The average of {numbers[0]} and {numbers[1]} is {average}")
```

2.

```
[kingvcheese@cachyos-x8664 school]$ python3 03\ Strings/exercise-02.py
Please enter two numbers: 13 13
Please enter two numbers
[kingvcheese@cachyos-x8664 school]$ python3 03\ Strings/exercise-02.py
Please enter two numbers: 22 1
The average of 22 and 1 is 11.5
```

```
# Prompt user to enter a key number between 1 and 26
# output the original string and encrypted string

print(r"""
_____
\      \      \      \      \      \      \      \      \      \
/      \      \      \      \      \      \      \      \      \
\      \      \      \      \      \      \      \      \      \
\      \      \      \      \      \      \      \      \      \
_____ (_____ /_____ >_____ >(_____ /_____
          \      \      \      \      \      \
          """)

original = """
If he had anything confidential to say,
he wrote it in cipher,
that is, by so changing the order of the letters of the alphabet,
that not a word could be made out.
"""

key = int(input("Please enter a key number between 1 and 26: "))

if key < 1 or key > 26:
    print("Please enter a key number between 1 and 26")
    exit()

encrypted = ""

for i in range(len(original)):
    if original[i].isalpha():
        if original[i].isupper():
            encrypted += chr((ord(original[i].lower()) - ord('a') + key) % 26 + ord('A'))
        else:
            encrypted += chr((ord(original[i].lower()) - ord('a') + key) % 26 + ord('a'))
    else:
        encrypted += original[i]

print(f"The original string is: {original}")
print(f"\nThe encrypted string is: {encrypted}")
```

3.

```
[kingvcheese@cachyos-x8664 school]$ python3 03\ Strings/exercise-03.py
```

[illegible]

```
Please enter a key number between 1 and 26: 12
```

The original string is:

If he had anything confidential to say,

he wrote it in cipher,

that is, by so changing the order of the letters of the alphabet, that not a word could be made out.

The encrypted string is:

Ur tq tmp mzkftuzs oazrupqzfumx fa emk,

tq idafq uf uz oubtqd,

ftmf ue, nk ea otmzsuzs ftq adpgd ar ftq xqffqde ar ftq mxbtmngf,

ftmf zaf m iadp oagxp nq ympq agf.

EXERCISE 04: BASIC FUNCTION

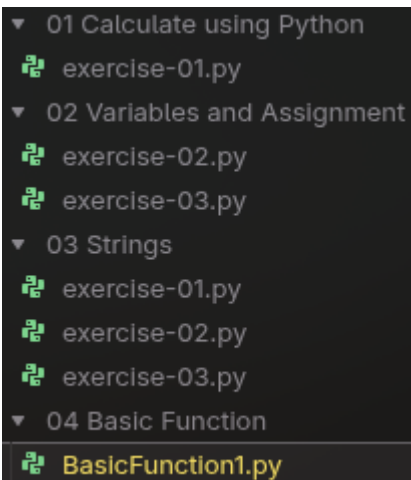
We are now saving each exercise.

1. Open IDLE and choose New File/Window and save as BasicFunction1.py
2. Write separate functions to do the following:

Function 1	Write a function to add two numbers together and print the answer. Use the following variable names and values: x = 17 y = 22
Function 2	Write a function to multiply x and y together and divide by z. Use the following variables in your function: x = 6 y = 4 z = 8

For each function:

- a. Ensure you have a docstring
 - b. Use at least one one-line comment
3. Call each function and ensure that it works



```
▼ 01 Calculate using Python
  exercise-01.py
▼ 02 Variables and Assignment
  exercise-02.py
  exercise-03.py
▼ 03 Strings
  exercise-01.py
  exercise-02.py
  exercise-03.py
▼ 04 Basic Function
  BasicFunction1.py
```

1.

```

def add(a, b):
    """ add a and b and print the output """
    print(a + b) # print the result

x = 17
y = 22

add(x, y)

def multiply_divide(a, b, z):
    """ multiply a and b and divide by z """
    return(a * b / z) # return the result

x = 6
y = 4
z = 8
❖
print(multiply_divide(x, y, z))

```

2.

```

[kingvchees
.py
39
3.0

```

3.

EXERCISE 05: EXTEND BASIC FUNCTION

1. Develop Function 2 from the previous exercise to use parameters and supply the following arguments:

x = 15

y = 13

z = 5

2. Write a function that will take in any string and print it out followed by 'You're welcome'. Supply the following argument to the function:

'There are only 10 types of people in the world.'

1.

```
def multiply_divide(a, b, z):  
    """ multiply a and b and divide by z """  
    return(a * b / z) # return the result  
  
x = 15  
y = 13  
z = 5
```

2.

```
def thanks(inp):  
    """ print a string and follow it by 'You're welcome' """  
    print(inp + " You're welcome")  
  
thanks("There are only 10 types of people in the world.")  
  
There are only 10 types of people in the world. You're welcome
```

EXERCISE 06: RETURNING VALUES

Write functions that will:

- Ask for the student name and return it.
- Ask for the name of their Computer Science teacher and return it.
- Ask for marks out of 10 for the last four homework tasks and return an average mark. *Hint: you will need to calculate the average inside the function.*
- Display the following message, depending on the average mark:
 - average ≥ 8 . 'Well done, X, Y is very pleased with your effort'
 - average $\geq 6 < 8$. 'A good effort, X.Y thinks you should check your work carefully'
 - average ≤ 5 . 'X this is very poor. Y has asked you to try harder'

Hint: X is the name of the student; Y is the name of the teacher. Remember to call functions correctly and declare variables being returned from each function.

```
What is your name? Uthman
What is your CS teacher's name? Mr Khostwal
What is the score for assignment 1? 8
What is the score for assignment 2? 9
What is the score for assignment 3? 1
What is the score for assignment 4? 7
A good effort Uthman, Mr Khostwal thinks you should check your work carefully
```

```
def get_name():
    """ ask for a name and return it """
    name = input("What is your name? ")
    return name

def get_cs_teacher_name():
    """ ask for a students cs teacher's name and return it """
    name = input("What is your CS teacher's name? ")
    return name

def average_homework():
    """ get the score out of 10 for the last 4 homework assignments and average them """
    test_scores = []
    for i in range(4):
        score = int(input("What is the score for assignment " + str(i+1) + "? "))
        test_scores.append(score)

    return sum(test_scores)/4

name = get_name()
teacher_name = get_cs_teacher_name()
average = average_homework()

# if average >= 8: well done {name}, {teacher_name} is very pleased with your effort
# if average >= 6 > 8: A good effort {name}, {teacher_name} thinks you should check your work carefully
# if average >= 5: {name} this is very poor, {teacher_name} has asked you to try harder

if average >= 8:
    print(f"Well done {name}, {teacher_name} is very pleased with your effort")
elif average >= 6:
    print(f"A good effort {name}, {teacher_name} thinks you should check your work carefully")
elif average >= 5:
    print(f"{name} this is very poor, {teacher_name} has asked you to try harder")
```

EXERCISE 07: RELATIONAL AND LOGICAL OPERATORS

Use print statements to find the answers to the following:

1. Calculate the following (True or False)
 - a. $23!=15$
 - b. $5+3<10$
 - c. $6 > 10 == 10 < 2$
2. If $a = 3$ and $b = 8$, what are the results of the following statements?
 - a. $a < b$
 - b. $6 >= a$
 - c. $b > a == \text{False}$
 - d. $\text{True} != (a == b)$
3. If $c = \text{True}$ and $d = \text{False}$, what are the results of the following statements?
 - a. $c \text{ and } d$
 - b. $\text{not } d \text{ or } c$
 - c. $c == d \text{ and True}$

```
print(23!=15)
print(5+3<10)
print(6>10==10<2)
```

1.

```
True
True
False
```

2. True
True
False
True

```
c = True
d = False

print(c and d)
print(not c or d)
print([c == d and True])
```

3.

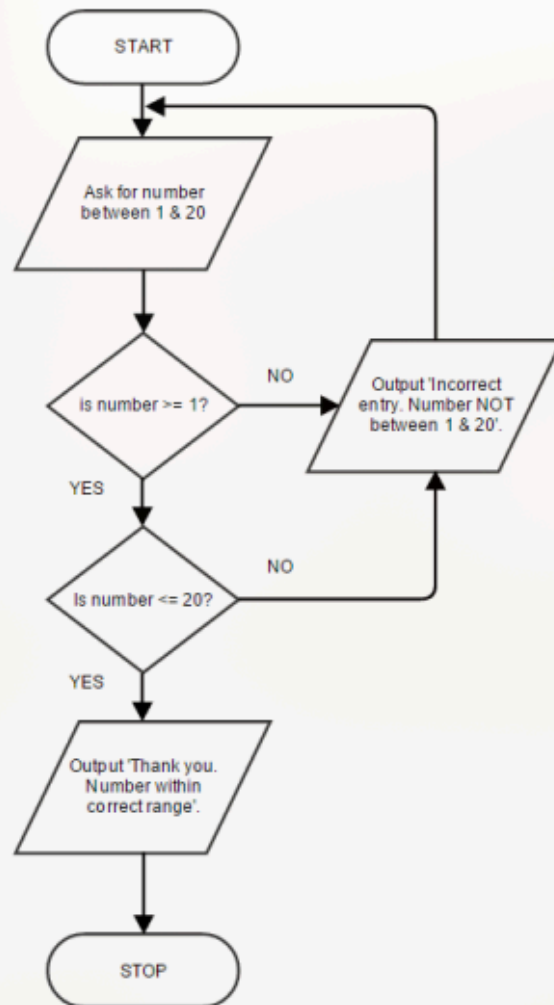
```
False
False
False
```

EXERCISE 08: IF/ELIF STATEMENTS

1. Complete the program shown in the flow chart.
2. Write a **program** which asks for the names of two football teams playing against each other and their scores.

Your program should calculate how many points each team gets (3 for a win, 1 for a draw, 0 if they lose).

Hint: you will need to declare constants to use in calculating the points. Remember, one team will be the HOME team, the other the AWAY team.



```
running = True
while running:
    num = int(input("Enter a number between 1 and 20: "))
    if num < 1:
        print("Incorrect entry. Number NOT between 1 & 20.")
        continue
    elif num > 20:
        print("Incorrect entry. Number NOT between 1 & 20.")
        continue
    print("Thank you. Number within correct range.")
    running = False
```

1.

```

# get names of two football teams and their scores
# 3 points if they win, 1 for draw, 0 for loss

WIN = 3
DRAW = 1
LOSS = 0

team1 = input("Enter the name of the first team: ")
team2 = input("Enter the name of the second team: ")
score1 = int(input("Enter the score of the first team: "))
score2 = int(input("Enter the score of the second team: "))

if score1 > score2:
    print(f"{team1} wins with a score of {WIN}")
    print(f"{team2} loses with a score of {LOSS}")
elif score1 < score2:
    print(f"{team2} wins with a score of {WIN}")
    print(f"{team1} loses with a score of {LOSS}")
else:
    print(f"The teams tied with a score of {DRAW}")

```

2.

```

[kingvcheese@cachyos-x8664 school]$ python3 08\ IF\&ELIF\ Statements/exercise-02.py
Enter the name of the first team: Home
Enter the name of the second team: Away
Enter the score of the first team: 3
Enter the score of the second team: 11
Away wins with a score of 3
Home loses with a score of 0
[kingvcheese@cachyos-x8664 school]$ python3 08\ IF\&ELIF\ Statements/exercise-02.py
Enter the name of the first team: Home
Enter the name of the second team: Away
Enter the score of the first team: 11
Enter the score of the second team: 3
Home wins with a score of 3
Away loses with a score of 0
[kingvcheese@cachyos-x8664 school]$ python3 08\ IF\&ELIF\ Statements/exercise-02.py
Enter the name of the first team: Home
Enter the name of the second team: Away
Enter the score of the first team: 3
Enter the score of the second team: 3
The teams tied with a score of 1

```

EXERCISE 09: WHILE AND FOR LOOPS

1. Write a program, using at least two functions AND parameter passing, which will allow the user to input a number between 1 and 12 and print the times table for that number.
2. Write a program, using at least 2 functions AND parameter passing, that will add together a series of numbers until the user enters 0. The program will then display the total.

```
# enter number between 1 and 12 print times tables

num = int(input("Enter a number between 1 and 12: "))

if num < 1 or num > 12:
    print("Incorrect entry. Number NOT between 1 & 12.")
    exit()

for i in range(1, 13):
    print(f"{num} x {i} = {num * i}")
```

1.

```
[kingvcheese@cachyos-x8664 school]$ python3 09\ Whi
Enter a number between 1 and 12: 3
3 x 1 = 3
3 x 2 = 6
3 x 3 = 9
3 x 4 = 12
3 x 5 = 15
3 x 6 = 18
3 x 7 = 21
3 x 8 = 24
3 x 9 = 27
3 x 10 = 30
3 x 11 = 33
3 x 12 = 36
[kingvcheese@cachyos-x8664 school]$ python3 09\ Whi
Enter a number between 1 and 12: 13
Incorrect entry. Number NOT between 1 & 12.
```

```

def calculate_sum():
    """Continuously prompts for numbers using a while loop until 0 is entered."""
    total = 0
    while True:
        num = float(input("Enter a number (or 0 to finish): "))
        if num == 0:
            break
        total += num
    return total

def display_total(total_value):
    """Displays the final calculated sum using parameter passing."""
    if total_value.is_integer():
        print(f"\nThe final total is: {int(total_value)}")
    else:
        print(f"\nThe final total is: {total_value}")

def main():
    final_sum = calculate_sum()
    display_total(final_sum)

if __name__ == "__main__":
    main()

```

2.

```

[kingvcheese@cachyos-x8664 school]$ py
Enter a number (or 0 to finish): 13
Enter a number (or 0 to finish): 3
Enter a number (or 0 to finish): 1
Enter a number (or 0 to finish): 0

The final total is: 17

```

EXERCISE 10: USING DATE AND TIME

Write a program that will display the day of the week that you were born using the import time module.

You will need to use functions to ask for the date, month and year (as integers). You should display the date to the user entered in this format, e.g. 15/Jan/97, and tell them what day of the week they were born.

```
import time

def get_birth_date():
    """Prompts the user for year, month, and day as integers."""
    year = int(input("Enter birth year (e.g., 1997): "))
    month = int(input("Enter birth month (1-12): "))
    day = int(input("Enter birth day (1-31): "))
    return year, month, day

def display_birth_info(year, month, day):
    """Converts the input integers to a time structure and displays formatted output."""
    date_tuple = (year, month, day, 0, 0, 0, 0, 0, -1)

    parsed_time = time.struct_time(date_tuple)

    seconds = time.mktime(parsed_time)
    formatted_struct = time.localtime(seconds)

    # Format date string as DD/Mon/YY (e.g., 15/Jan/97)
    formatted_date = time.strftime("%d/%b/%y", formatted_struct)

    day_of_week = time.strftime("%A", formatted_struct)

    print(f"\nDate entered: {formatted_date}")
    print(f"You were born on a {day_of_week}.")

year, month, day = get_birth_date()
display_birth_info(year, month, day)
```

```
Enter birth year (e.g., 1997): 2011
Enter birth month (1-12): 12
Enter birth day (1-31): 14

Date entered: 14/Dec/11
You were born on a Wednesday.
```

EXERCISE II: RANDOM OPTIONS

Write a program for a restaurant where the customers are given a random meal.

You will need lists for the starters, main course, desserts and drinks, with at least four items in each of your choices.

The program must generate a random meal with a choice from each menu option and display this to the customer like this:

```
>>>
*****
                MENU
Starter: Tomato soup
Main : Fish and Chips
Dessert: Mango Surpise
Drinks: Cup of Coffee
*****
```

```
STARTERS = ["Tomato Soup", "Baked Camembert", "Prawn Cocktail", "Garlic Butter Mushrooms"]
MAINS = ["Chicken Tikka Masala", "Beef Stew", "Chicken Parmesan", "Chicken Curry"]
DESSERTS = ["Chocolate Cake", "Ice Cream", "Lemon Meringue Pie", "Cheesecake"]
DRINKS = ["Coffee", "Tea", "Water", "Lemonade"]
```

```
import random
```

```
starter = random.choice(STARTERS)
main = random.choice(MAINS)
dessert = random.choice(DESSERTS)
drink = random.choice(DRINKS)
```

```
print("*****")
print("        MENU")
print(f"Starter: {starter}")
print(f"Main : {main}")
print(f"Dessert: {dessert}")
print(f"Drinks : {drink}")
print("*****")
```

```
*****
                MENU
Starter: Baked Camembert
Main : Chicken Tikka Masala
Dessert: Chocolate Cake
Drinks : Lemonade
*****
```

EXERCISE 12: CREATE A SHAPE

Write the code to create:

- a regular octagon in red
- a regular nonagon in blue

In your answer:

- a) Use condition-controlled iteration
- b) Both shapes must be displayed on the same screen

Note: use any suitable names for your screen and turtle variables.

```
import turtle

my_screen = turtle.Screen()
terry = turtle.Turtle()

# Regular octagon in red
terry.pencolor("red")
terry.pendown()

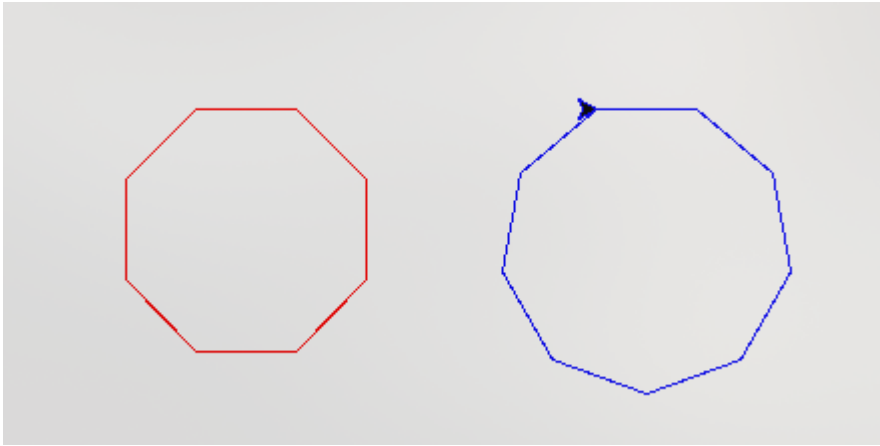
count = 0
while count < 8:
    terry.forward(50)
    terry.right(45) # 360 / 8
    count += 1

# Move turtle so shapes don't overlap
terry.penup()
terry.forward(200)

# Regular nonagon in blue
terry.pencolor("blue")
terry.pendown()

count = 0
while count < 9:
    terry.forward(50)
    terry.right(40) # 360 / 9
    count += 1

my_screen.exitonclick()
```



EXERCISE 13: SLICES

1. Open the Python shell and create a list of the months of the year; you can use abbreviations like 'Jan', 'Feb', etc.
2. SLICE the list to create these results:

```
>>>
['Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug']
['Jan', 'Feb', 'Mar', 'Apr']
['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul', 'Aug', 'Sep', 'Oct', 'Nov']
['Jul', 'Aug', 'Sep', 'Oct', 'Nov', 'Dec']
```

3. What happens if you type:

```
months[:-6]
months[-3:]
```

(or whatever you have called your list of months)?

4. What have you learned from this?

1.

```
>>> months = ["Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul", "Aug", "Sep", "Oct", "Nov", "Dec"]
```
- 2.

```
>>> months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December']
>>> print(months[2:8])
['March', 'April', 'May', 'June', 'July', 'August']
>>> print(months[:4])
['January', 'February', 'March', 'April']
>>> print(months[:-1])
['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November']
>>> print(months[-6:])
['July', 'August', 'September', 'October', 'November', 'December']
>>> print(months[:-6])
['January', 'February', 'March', 'April', 'May', 'June']
>>> print(months[-3:])
['October', 'November', 'December']
```

- 3.
4. I have learned python uses start stop syntax [start:stop], you can leave the start or stop empty, and you can use negatives to index backwards (right to left, not left to right)

```

>>> friends_1 = ['Sam', 'Billy', 'Jamie', 'Danny', 'Emily']
>>> friends_2 = ['Claire', 'Jenny', 'Charlie', 'Tom', 'Aaron']
>>> all_friends = friends_1 + friends_2
>>> all_friends
['Sam', 'Billy', 'Jamie', 'Danny', 'Emily', 'Claire', 'Jenny', 'Charlie',
'Tom', 'Aaron']
>>> all_friends.sort()
>>> all_friends
['Aaron', 'Billy', 'Charlie', 'Claire', 'Danny', 'Emily', 'Jamie', 'Jenny',
'Sam', 'Tom']

```

EXERCISE 14: PRACTISE IN THE SHELL

1. Open the Python Shell and try each of the examples above; you do not have to save anything.
2. Make sure that each example works as shown.

```

friends_1 = ['Sam', 'Billy', 'Jamie', 'Danny', 'Emily']
✓

friend_2 = ['Claire', 'Jenny', 'Charlie', 'Tom', 'Aaron']
✓

all_friends = friends_1 + friend_2
✓

all_friends.sort()
✓

all_frie
⊗
Traceback (most recent call last):
  File "/home/kingvcheese/.vscode/extensions/ms-python.python-2026.4.0-linux-x64/python_files/python_server.py", line 139, in exec_user_input
    retval = callable(user_input, user_globals)
  File "<string>", line 1, in <module>
NameError: name 'all_frie' is not defined. Did you mean: 'all_friends'?

all_friends
✓
['Aaron', 'Billy', 'Charlie', 'Claire', 'Danny', 'Emily', 'Jamie', 'Jenny', 'Sam', 'Tom']

```

```
>>> students = ['Tom', 'Leah', 'Jamie', 'Holly', 'Charlie', 'Aaron']
>>> students.reverse()
>>> students
['Aaron', 'Charlie', 'Holly', 'Jamie', 'Leah', 'Tom']
```

EXERCISE 15: PRACTICE IN THE PYTHON SHELL

1. Open the Python Shell and try each of the examples above; you do not have to save anything.
2. Make sure that each example works as shown.

```
students = ['Tom', 'Leah', 'Jamie', 'Holly', 'Charlie', 'Aaron']
```

✓

```
students.reverse()
```

✓

```
students
```

✓

```
['Aaron', 'Charlie', 'Holly', 'Jamie', 'Leah', 'Tom']
```

EXERCISE 19: DEVELOP THE FAVOURITE FOODS PROGRAM

You will use the program to find out the favourite meals of students in your class.

1. Use the example here as a starting point; your program also needs to ask for each student's name and add this to each row of data in the 2D list.
2. Your program should ask whether more data is to be added after each student's details are entered.
3. Your program should print the full 2D list of data collected in a tabular format when all data has been added.

```
# Initialize as a clean empty list
foods = []

run = True
while run:
    name = input("Enter a name: ")
    food = input("Enter a favourite food: ")
    foods.append([name, food])

    user_choice = input("Would you like to add another student? (y/n): ")
    if user_choice.strip().lower() == "n":
        run = False

# Print in tabular format
print("\nName - Favourite Food")
print("-" * 25)
for person in foods:
    print(f"{person[0]} - {person[1]}")
```

```
Enter a name: Uthman
Enter a favourite food: Cheese
Would you like to add another student? (y/n): y
Enter a name: Bojan
Enter a favourite food: Pancakes
Would you like to add another student? (y/n): n

Name - Favourite Food
-----
Uthman - Cheese
Bojan - Pancakes
```

EXERCISE 20: FUNDRAISER PART 1

It can be very useful to use a 2D list to store data for your program so that data can be passed easily between functions, making your code more efficient. For example:

Activity	Charge per hour
Bag Pack	£2.50
Car Wash	£3.50



Your school is organising a charity fundraising day and wants students to volunteer to take part in an activity. As part of this, you think it will be useful to try to work out how much money students in your form could raise by deciding on a charge for the activities. The three activities your form has chosen are:

- Bag packing at a local supermarket
- Washing cars at school
- Dog walking

Tasks

1. Write a program that will display a table of activities with the charge per hour, similar to the one shown on the previous page.
2. The program should:
 - a. Ask for the name of the activity.
 - i. For extra credit, the maximum length should be 10 characters or fewer.
 - ii. You should display a suitable message if the text length is longer.
 - iii. Your code should not allow the user to continue until a suitable length of characters has been entered.
 - b. Ask for the amount to be charged per hour for the activity.
 - i. For extra credit, the acceptable charges are between 99p and £5.99.
 - ii. You should display a suitable message if the charge is not within these values.
 - iii. Your code should not allow the user to continue until a value between those amounts has been entered.
3. When all data has been entered, the program should print out the information in a tabular format with suitable column headings.

Your solution should use:

- Functions and a main() function to run the program.
- The functions should have the correct structure and use docstrings to say what they do.
- You should pass the data structure between functions in order to print it.
- You could consider using helper functions for more efficient code (see Add User Input to a 2D List).

```

# loop:
# ask for name max length 10 (retry until suitable)
# ask for amount (hourly) (between 99p and £5.99) (retry until suitable)
# after user confirms they are done entering data,
# print out the data in tabular format with suitable column headings
# use main() function and functions with docstrings

def get_name():
    collect = True
    while collect:
        name = input("Enter the activity: ")
        if len(name) > 10:
            print("Name too long. Please try again.")
            continue
        else:
            collect = False
    return name

def get_amount():
    collect = True
    while collect:
        amount = float(input("Enter activity hourly rate in pence: "))
        # if less than 99 pence or greater than 599
        if amount < 99 or amount > 599:
            print("Amount out of range. Please try again.")
            continue
        else:
            collect = False
    return amount

def print_data(data):
    print("\nActivity - Amount")
    print("-" * 15)
    for activity in data:
        print(f"{activity[0]} - {f'£{activity[1]/100}'")

def main():
    run = True

    data = []

    while run:
        activity = get_name()
        amount = get_amount()
        print(f"{activity} - {f'£{amount/100}'")
        user_choice = input("Would you like to add another activity? (y/n): ")
        if user_choice.strip().lower() == "n":
            run = False

        data.append([activity, amount])

    print_data(data)

if __name__ == "__main__":
    main()

```

```
Enter the activity: Hiking
Enter activity hourly rate in pence: 599
Hiking - £5.99
Would you like to add another activity? (y/n): y
Enter the activity: Baking
Enter activity hourly rate in pence: 99
Baking - £0.99
Would you like to add another activity? (y/n): y
Enter the activity: Cycling
Enter activity hourly rate in pence: 24
Amount out of range. Please try again.
Enter activity hourly rate in pence: 699
Amount out of range. Please try again.
Enter activity hourly rate in pence: 399
Cycling - £3.99
Would you like to add another activity? (y/n): n

Activity - Amount
-----
Hiking - £5.99
Baking - £0.99
Cycling - £3.99
```

EXERCISE 21: FUNDRAISER PART 2

In this part of the exercise, you need to enter the student name, ask for the activity they completed, ask how many hours they spent on the activity, and calculate the totals for each student.

1. Develop the program to ask for the number of students you want to enter.
2. Using helper functions, the program should:
 - a. Ask for the student name.
 - b. Ask for the activity completed (using the list created in the previous exercise).
 - c. Ask for the number of hours spent on the activity (as a whole number).
3. Your program should then use the values entered for each activity (from the list created in the previous exercise) to calculate a total collected.
4. Your program should then print out the information about student name, activity completed, hours spent and total funds each student has raised.

Please see [here](#) for code

```
Enter the activity: dance
Enter activity hourly rate in pence: 399
Would you like to add another activity? (y/n): n

Activity - Amount
-----
dance - £3.99

How many students do you want to enter? 2
Enter student name: Uthman
Available activities: dance
Enter the activity this student did: dance
Enter number of hours spent (whole number): 3
Enter student name: Abubakar
Available activities: dance
Enter the activity this student did: 2
Not a valid activity. Please try again.
Available activities: dance
Enter the activity this student did: dance
Enter number of hours spent (whole number): 2

Student Name - Activity - Hours - Total Raised
-----
Uthman - dance - 3 - £11.97
Abubakar - dance - 2 - £7.98
```

EXERCISE 22: DEVELOP THE SIMPLE GAME BOARD

1. Write a function to create a game board grid, measuring 8×8 .
2. Add a row of numbers, from 1 to 8, across the top of the grid.
 - a. Hint: this will be the first nested list in your game board list.
3. Add your player location in the bottom left-hand corner of the grid.
4. Create a function to print out the grid.
5. Call your two functions from within a main function.

```
create_grid, print_grid, main = lambda: """ """ ; print('Grid for 4 in range(1, 9):') ; for i in range(1, 9): ; print('Row %d:' % i) ; print('') ; print('') ; if (i, 0) == (7, 0): ; print('X' if (i%2 == 0 else 'O') ; for c in range(8): ; for i in range(8): ; lambda grid: ; print(grid) ; lambda print_grid: ; lambda print_grid: ; main() ;
```

^ code ^

```
1 2 3 4 5 6 7 8
1 X O X O X O X O
2 O X O X O X O X
3 X O X O X O X O
4 O X O X O X O X
5 X O X O X O X O
6 O X O X O X O X
7 X O X O X O X O
8 * X O X O X O X
```

EXERCISE 23: MOVE A PLAYER PART 1

Using the file you have created for the Simple Board Game:

1. Write a function to ask if the player wants to move up, move down or stay on the row. Make sure you store their response for use later.
2. Write a function to ask whether the player wants to move left, move right or stay on the same column. Make sure you store their response for use later.
3. Write a function to ask how many 'squares' the player wants to move. Again, you should store this response in a suitable variable for use later.

Please see [here](#) for code

```
> python3 23\ Move\ a\ player\ part\ 1\exercise-01.py
Enter direction and magnitude (u, d, l, r) (q to quit): u 3
  0  1  2  3  4  5  6  7
0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0
2  0  0  0  0  0  0  0
3  0  0  0  0  0  0  0
4  *  0  0  0  0  0  0
5  0  0  0  0  0  0  0
6  0  0  0  0  0  0  0
7  0  0  0  0  0  0  0

Enter direction and magnitude (u, d, l, r) (q to quit): r 1
  0  1  2  3  4  5  6  7
0  0  0  0  0  0  0  0
1  0  0  0  0  0  0  0
2  0  0  0  0  0  0  0
3  0  0  0  0  0  0  0
4  0  *  0  0  0  0  0
5  0  0  0  0  0  0  0
6  0  0  0  0  0  0  0
7  0  0  0  0  0  0  0

Enter direction and magnitude (u, d, l, r) (q to quit): 8
Invalid magnitude.
Enter direction and magnitude (u, d, l, r) (q to quit): u 8
Invalid Y coordinate
Please dont go out of bounds
Enter direction and magnitude (u, d, l, r) (q to quit): q
```

EXERCISE 24: MOVE A PLAYER PART 2

You should now know enough to be able to ask which direction to move and how many squares to move, and make the game board show this move.

1. Write a function to make a move, inside which you should:
 - a. Call the function for vertical movement
 - b. Call the function for the number of squares to move
 - c. Check whether this is possible by using the current player location for axis 'y'
 - d. If this is not a valid move: the game is over
 - e. If the move is possible
 - f. Call the function for horizontal movement
 - g. Call the function for the number of squares to move
 - h. Check the move is possible by using the current player location for axis 'x'
 - i. If this is not a valid move: the game is over
 - j. If the move is possible
 - k. Return the new x and y values
 - l. Update the previous player location to show 'O' not 'X'
 - m. Print the updated grid with the new player location
2. Add a while loop to the main function to keep asking/moving until five moves have been made.

TIP: In order to update the board and change the 'X' to a 'O' when a move is made, you should store the current x and y values in two new variables.

Please see [here](#) for code

EXERCISE 25: CREATE A MAGIC SQUARE

An easy version of this game will be to create the magic square location just once and then check whether the player has moved to that location each time the game loop runs.

The hard version of the game involves creating a new random magic square each time the game loop runs.

1. Write your function to create the location of the magic square.
2. Write a function that will ask the player whether they want to continue playing the game; ensure that you write robust code to check their answer can only be Y or N.
3. Develop the while loop that runs your game so that the game continues until the answer is N.

Please see [here](#) for code

EXERCISE 26: DICTIONARIES

Write a program which uses at least two functions. The program should:

1. Create a dictionary with key/value pairings for revision of a subject of your choice, e.g. a glossary for computer science terms. Aim for at least six pairs in the dictionary.
2. The program should display the value and ask for the key, as in the example above.
3. The program must keep track of the correct answers and display the score, as a percentage, when all terms have been answered.
4. To extend this, your program should now keep asking for the definitions until the user has scored 60% or more.

```
animals = {
    "cat": "meow",
    "dog": "woof",
    "bird": "chirp",
    "fish": "swim",
    "snake": "slither",
    "horse": "neigh",
}

score = [0, 0] # correct, total

run = True

while run:
    for i in animals:
        print(f"What animal goes {animals[i]}?")
        answer = input("> ")

        if answer.lower().strip() == i.lower():
            print(f"Correct! {i} goes {animals[i]}.")
            score[0] += 1
        score[1] += 1
    print(f"Score: {round(score[0]/score[1]*100)}%")

    if not round(score[0]/score[1]*100) < 60:
        run = True
    score = [0, 0]
```

EXERCISE 27: NESTED DICTIONARY

Write a program that will use at least two functions. The program should:

1. Create a dictionary of student names and their homework scores out of 20.
2. Allow the teacher to add a student name and their homework marks.
3. Print out all records.

Test your program by entering the following sample data:

Name	HW1	HW2	HW3	HW4
James Farrow	15	12	18	11
Jenny Sullivan	8	12	13	16
Gemma Thompson	14	16	13	15
Oliver Booth	17	15	18	16
Kelly Jones	13	12	18	9

```

student_names = {}

# data structure
# student_names = {
#     "john": {
#         "HW1": 15
#         "HW2": 20
#     },
#     "jane": {
#         "HW1": 10
#         "HW2": 15
#     }
# }

def get_student_name():
    while True:
        student_name = input("Enter student name: ").strip().lower()
        if student_name == "!break":
            return None
        elif student_name in student_names:
            return student_name
        else:
            return student_name

def update_data(student_name, hw, score):
    if student_name not in student_names:
        student_names[student_name] = {}

    if hw not in student_names[student_name]:
        student_names[student_name][hw] = score
    else:
        print("HW already exists.")

def display_data():
    for student in student_names:
        print(f"{student}:")
        for hw in student_names[student]:
            print(f"    {hw}: {student_names[student][hw]}")

def main():
    Run = True

    while Run:
        student_name = get_student_name()

        if student_name is None:
            Run = False
            print("Outputting data...")
            break

        hw = input("Enter homework name: ").strip().lower()

        score = int(input("Enter score: "))
        update_data(student_name, hw, score)

        display_data()

if __name__ == "__main__":
    main()

```

EXERCISE 28: FILE HANDLING

1. Create a text file with the first names of 12 students in your class and save it as 'students.txt'. Read in the file into a list and sort it into order, write the list back to the file and check the names are now in alphabetical order.
2. Create a text file with some items of grocery shopping, save as 'shopping.txt'. Write a program that will:
 - a. Read in the file into a suitable data structure
 - b. Ask the user for three additional items to add to the list
 - c. Check whether the item is in the list, if not add it to the list
 - d. Print out the new list with the three additional items

```
students = [  
    "john",  
    "jane",  
    "bob",  
    "alice",  
    "charlie",  
    "david",  
    "emily",  
    "frank",  
    "grace",  
    "harry",  
    "isabella",  
    "jessica",  
]  
  
with open("students.txt", "w") as f:  
    for student in students:  
        f.write(f"{student}\n")  
  
students = []  
  
with open("students.txt", "r") as f:  
    for line in f:  
        students.append(line.strip())  
  
print("Unsorted: ")  
print(students)  
  
students.sort()  
print(" ")  
  
print("Sorted: ")  
print(students)  
  
with open("students.txt", "w") as f:  
    for student in students:  
        f.write(f"{student}\n")
```

1.

```
groceries = []

with open("shopping.txt", "r") as f:
    for line in f:
        groceries.append(line.strip())

print(groceries)

for i in range(3):
    item = input("Enter item to add to shopping list: ").strip().lower()
    groceries.append(item)

print(groceries)
```

2.

EXERCISE 29: VALIDATION

1. The code shown on the right should only accept data in the range 0–100.

Amend the code so that the user cannot continue until a value in the correct range has been entered.

Your code must display an error message if a value outside this range is entered.

```
def get_score():
    """asks for test score"""
    test_score = int(input('Enter test score: '))
    return test_score

def main():
    """runs all functions"""
    test_score = get_score()
    print("These are the GCSE grade boundaries:")
    if test_score >= 90:
        print('Your grade is 9')
    elif test_score >= 80:
        print('Your grade is 8')
    elif test_score >= 70:
        print('Your grade is 7')
    elif test_score >= 60:
        print('Your grade is 6')
    elif test_score >= 50:
        print('Your grade is 5')
    elif test_score >= 40:
        print('Your grade is 4')
    elif test_score >= 30:
        print('Your grade is 3')
    elif test_score >= 20:
        print('Your grade is 2')
    else:
        print('Your grade is 1')

main()
```

```

def get_score():
    """ asks for test score """
    collect = True

    while collect:
        test_score = input("Enter test score: ").strip()
        if test_score.isnumeric():
            test_score = int(test_score)
            if test_score >= 0 and test_score <= 100:
                collect = False
            else:
                print("Invalid input. Please enter a number between 0 and 100.")
        else:
            print("Invalid input. Please enter a number.")

    return test_score

def main():
    """ runs all functions """
    test_score = int(get_score())

    print("These are the GCSE grade boundaries:")
    if test_score >= 90:
        print("Your grade is 9")
    elif test_score >= 80:
        print("Your grade is 8")
    elif test_score >= 70:
        print("Your grade is 7")
    elif test_score >= 60:
        print("Your grade is 6")
    elif test_score >= 50:
        print("Your grade is 5")
    elif test_score >= 40:
        print("Your grade is 4")
    elif test_score >= 30:
        print("Your grade is 3")
    elif test_score >= 20:
        print("Your grade is 2")
    else:
        print("Your grade is 1")

main()

```

EXERCISE 30: AUTHENTICATION

Part A: The authentication code example using 2D arrays checks that the username is entered correctly after three attempts; however, it only gives the user one attempt at the password.

Improve the code, using functions and parameter passing, so that each part of the process allows the user three attempts before displaying the message on Line 31.

Part B: The current authentication code does not allow for new users to be added or any former users to be removed, which is clearly a security issue.

Improve your solution to incorporate a method to ensure that the list of system users can be updated without needing to edit the original program code.

1. Please see [here](#) for code
2. Please see [here](#) for code

EXERCISE 31: ERROR HANDLING

1. What type of error is shown in this code snippet?

```
number = 1
while number < 13:
    print("{0} squared = {1}".format(number, number * number))
```

2. Correct the code so that it will print the squared numbers from 1 to 12.
3. Rewrite the code below so that there are no exception errors when trying to open this non-existent file.

```
def writeFile():
    """write data to a file"""
    myFile = open('new_file.txt', 'r')
    myFile.write("Little Bo Peep has lost her sheep ")
    myFile.write("\n")
    myFile.write("And doesn't know where to find them.")
    myFile.write("\n")
    myFile.close()
```

1. IndexError

```
number = 1
while number < 13:
    print("[{0} squared = {1}].format(number, number * number))
    number += 1
```

- 2.

EXERCISE 32: DATABASES

1. What data will be in the school database about each student?
2. What data could be used to uniquely identify each student?

1. Birth date, Form, Timetable, Consequences, Reward points, Detentions, Medical Notes, Parent phone number, email address, password (hashed), username.
2. Name, Parent phone number, email address, username, id number/uuid

EXERCISE 33: PRIMARY KEYS

1. Identify the PRIMARY KEY in the original employee table.
2. The school is creating a database of staff cars to ensure that only authorised staff can park on the school site.
 - a. Write down five possible fields in the cars table.
 - b. State which field would be the primary key and give a reason.

1. EmployeeID
2. CarRegistration, EmployeeID, Make, Colour, PermitExpiryDate
3. CarRegistration as it is unique and does not change.

EXERCISE 34: DATA TYPES

Look at the Python code used to create the 'games' table on the following page.

1. Which field is the most suitable as a primary key?
2. What data type is used in this field?
3. Explain why you have chosen this field.
4. Identify the data type used in the 'price' field.
5. Explain why you think this data type has been chosen.

1. code.
2. Integer.
3. Code is suitable as the game needs a unique identifier.
4. Money.
5. Price is the cost of a game so values can be represented as currency.

EXERCISE 35: QBE GRIDS

Using the query by example templates at the back of this resource:

1. Write a query that will find all games for which the cost is less than £20.
2. Write a query that will select the game where the code equals 'arkhr01' and print out the title and price only.

1.

```
SELECT *  
FROM games  
WHERE price < 20;
```
2.

```
SELECT name, price  
FROM games  
WHERE code = 'arkhr01';
```

EXERCISE 36: DATABASE QUESTIONS

A database was set up to store details of books at a library. Below is a sample of the data.

Title	Author	Format	Pages	ISBN	PublicationDate
Drawing Manga	9colorstudio	Paperback	44	9781615645619	2013
Flashpoint	Johns, Geoff	Paperback	166	9781401233389	2012
House of M	Bendis, Brian	Paperback	224	9780785117216	2006
Punishermax: Homeless	Aaron, Jason	Paperback	128	9780785152101	2012
Punishermax: Frank	Aaron, Jason	Paperback	112	9780785152095	2012
Ripclaw	Aaron, Jason	Paperback	208	9781607060543	2009
Wallace & Gromit: The Bootiful Game	Rimmer, Ian	Paperback	48	9781840239485	2005

1. State the number of fields in each record.
2. Identify which field should be used as the primary key, and explain why.
3. State the title of the book(s) that are returned using the following search criteria:

(Name = * Jason) AND (PublicationDate < 2012)

1. 6
2. ISBN as its unique identifier
3. Ripclaw

EXERCISE 37: COMPARISON OPERATORS

1. Copy and complete the following table by explaining the meaning of the following *comparison operators* used in queries.

Operator	Meaning
=	
>	
<	
>=	
<=	
<>	

1. = equal to
> greater than
< less than
>= greater than or equal to
<= less than or equal to
<> not equal to

EXERCISE 38: DATABASE QUERIES

The following diagram shows part of a membership database.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0001	Kirkland	Haley	M	18	York	2014
0002	Foley	Thor	M	19	Birmingham	2015
0003	Sanders	Nomlanga	M	17	Manchester	2017
0004	Allen	Edan	M	20	Leeds	2015
0005	Brewer	Jane	F	16	London	2016
0006	Abbott	Eagan	M	18	Birmingham	2017
0007	Kline	Clara	M	18	Birmingham	2014
0008	Miles	Ivor	F	20	Bristol	2018
0009	Patrick	Gannon	M	16	York	2018
0010	Miranda	Jelani	M	17	Sheffield	2019
0011	Pena	Jelani	F	18	Sheffield	2014
0012	Austin	Cherokee	M	16	Bristol	2018
0013	Wilkins	Rafael	M	19	London	2014
0014	Cote	Briar	M	16	Manchester	2019
0015	Good	Josephine	F	16	Newcastle	2016
0016	Noble	Arthur	M	17	London	2018
0017	Good	Zachery	F	19	Leeds	2016
0018	Valencia	Solomon	F	18	York	2017
0019	Eaton	Brenden	M	17	London	2013
0020	Trevino	Andrew	M	16	Bristol	2016
0021	Mccarthy	Kalia	M	16	York	2015
0022	Beach	Beck	M	16	Bristol	2014
0023	David	Cora	M	17	York	2017
0024	Castillo	Kitra	F	18	Sheffield	2015
0025	Sawyer	Nomlanga	F	19	Manchester	2018
0026	Carver	Tyrone	M	16	Leeds	2019
0027	Bullock	Lisandra	M	19	Sheffield	2014
0028	Hurst	Camilla	F	20	Leeds	2016
0029	Mcfadden	Hannah	M	16	York	2015
0030	Mcintyre	Kareem	F	19	Birmingham	2015

For the following question:

State the search criteria which would have been input to produce the following results.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0002	Foley	Thor	M	19	Birmingham	2015
0004	Allen	Edan	M	20	Leeds	2015
0013	Wilkins	Rafael	M	19	London	2014
0027	Bullock	Lisandra	M	19	Sheffield	2014

The answer is: **Gender = 'M' AND Age >= 19**

Using the same membership database from the previous page, state the search criteria which would have been input to produce the following results:

1.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0004	Allen	Edan	M	20	Leeds	2015
0010	Miranda	Jelani	M	17	Sheffield	2019
0011	Pena	Jelani	F	18	Sheffield	2014
0017	Good	Zachery	F	19	Leeds	2016
0024	Castillo	Kitra	F	18	Sheffield	2015
0026	Carver	Tyrone	M	16	Leeds	2019
0027	Bullock	Lisandra	M	19	Sheffield	2014
0028	Hurst	Camilla	F	20	Leeds	2016

2.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0009	Patrick	Gannon	M	16	York	2018
0010	Miranda	Jelani	M	17	Sheffield	2019
0012	Austin	Cherokee	M	16	Bristol	2018
0014	Cote	Briar	M	16	Manchester	2019
0016	Noble	Arthur	M	17	London	2018
0026	Carver	Tyrone	M	16	Leeds	2019

3.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0017	Good	Zachery	F	19	Leeds	2016
0028	Hurst	Camilla	F	20	Leeds	2016

4.

Member ID	Surname	Forename	Gender	Age	City	Year Joined
0005	Brewer	Jane	F	16	London	2016
0008	Miles	Ivor	F	20	Bristol	2018
0011	Pena	Jelani	F	18	Sheffield	2014
0015	Good	Josephine	F	16	Newcastle	2016
0018	Valencia	Solomon	F	18	York	2017
0024	Castillo	Kitra	F	18	Sheffield	2015
0025	Sawyer	Nomlanga	F	19	Manchester	2018
0030	Mcintyre	Kareem	F	19	Birmingham	2015

1. City = 'Leeds' OR City = 'Sheffield'
2. Age < 18 AND YearJoined >= 2018
3. City = 'Leeds' AND YearJoined = 2016
4. Gender = 'F'

EXERCISE 39: MULTIPLE TABLE SEARCH

Write the SQL code to search for:

1. Title and Director for all DVDs with a category rating of 15.
2. Title and YearReleased for all DVDs in the 'comedy' genre.

1.

```
SELECT DVDs.Title, DVDs.Director
FROM DVDs, Category
WHERE DVDs.CategoryID = Category.CategoryID
AND Category = '15';
```
2.

```
SELECT DVDs.Title, DVDs.YearReleased
FROM DVDs, Genre
WHERE DVDs.GenreID = Genre.GenreID
AND Genre = 'comedy';
```

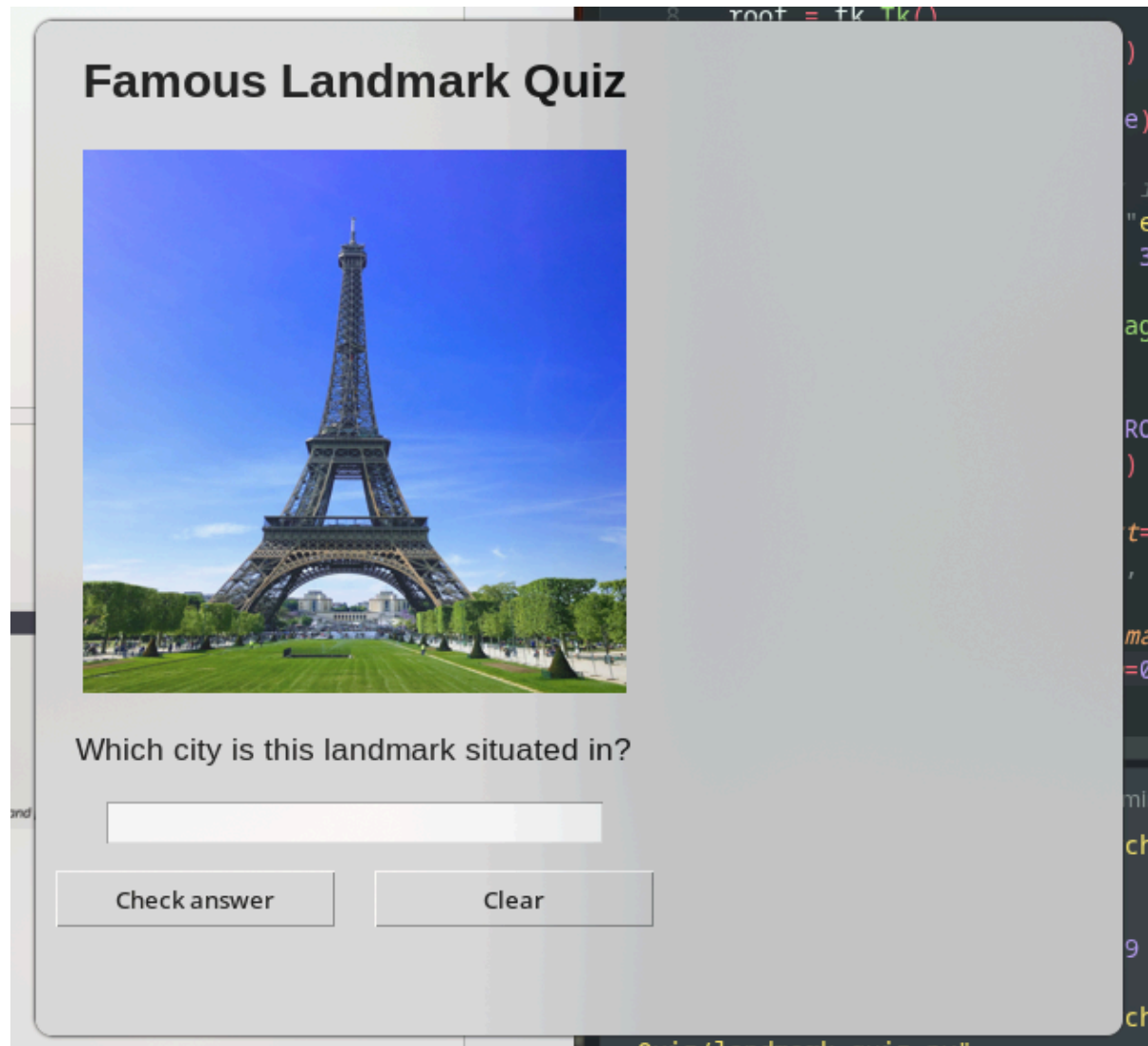
EXERCISE 40: SIMPLE GUI QUIZ

Create a GUI that will:

1. Include a suitable title and icon in the GUI window
2. Display a picture of a famous landmark, e.g. the Eiffel Tower
3. Ask the user to enter the name of the city where the landmark is situated
4. If the answer is correct, display suitable text message
5. If the answer is wrong, ask the user to try again

Hint: remember to store your images in the same folder as your Python file. You will need to amend row positions, column positions and padding to correctly display your GUI.

Please see [here](#) for code



EXERCISE 41: COMPLETE THE UPDATE FEATURE

The partially completed program now needs to be completed to include an UPDATE feature; one of your contacts may change their email, phone number or surname.

1. Add a function to perform this update using the OID field to identify the record
2. Attach the function to a new button and check it works

HINT: you will need to insert the field name into your UPDATE sql command but you do not know which the user will choose. The simplest method is to use the 'dot' format() string built-in function like this:

```
UPDATE database_name SET {0} = ? WHERE field_name = ?.format (update_col_name), (value1,value2)
```

For extra credit, improve the look or the functionality of this simple GUI.

Please see [here](#) for code

First Name

Surname

Mobile

Email

Save

Display all

Enter OID

Delete

Update field

New value

Update

OID: 1 | John Green | Mobile: 098762358 | Email: Green_m@hotmail.com
OID: 2 | Sally Jones | Mobile: 0987654321 | Email: sallyj@outlook.com

Practice assignment 1

Please see code [here](#)

 example.mp4

Practice assignment 2

Please see code [here](#)

Practice assignment 3

Please see code [here](#)

Practice assignment 4

Please see code [here](#)

Practice assignment 5

Please see code [here](#)