

Software Development Assignment James Frew 2108660 *by James Frew*

Submission date: 26-Apr-2022 03:48PM (UTC+0100)

Submission ID: 177362918

File name:

31653_James_Frew_Software_Development_Assignment_James_Frew_2108660_410175_1737625753.docx
(724.53K)

Word count: 3554

Character count: 17907

ACCB4017 - Software Development

James Frew 2108660

2108660@student.uwtsd.ac.uk

Table of Contents

Part A	3
A1	3
A2	3
A3	4
Part B	6
Exercise 1 – Data entry	6
Logical Stages	6
Code with annotation	6
Testing	7
Exercise 2 – Shape area calculator	10
Logical Stages	10
Code with annotation	10
Testing	12
Exercise 3 – Petrol cost.....	16
Logical Stages	16
Code with annotation	16
Testing	17
Exercise 4 – Pairs game	22
Logical Stages	22
Code with annotation	22
Testing	24

Part A

A1

Variables are values that are used to store data by a specific name and type. Types of variables include: Integers (whole numbers), Floats (decimal numbers), Strings (letters), Booleans (true or false), etc. Variables are important in programming as storing data is very important. When performing calculations, the calculation can be performed once, and a variable used to store the result of the calculation for later use. Without variables, the result would have to be calculated every time it was needed, rather than just calling the variables name where the data is stored. Additionally, if a value changes, the only change that has to be made to the program is to the variable, rather than every instance of where the variable is used.

In Python, variables can either be assigned like this:

```
integerVariable = 1
```

Or like this:

```
integerVariable = int(1)
```

The first example is that of a dynamic variable. Python is able to interpret what type a variable is without specifically telling it which it is. This doesn't always produce the required results, so variable types can be set manually as in the second example. The following are examples of performing calculations in python, first without variables, then with variables:

```
print("The result is:", 2 + 5)
print("Multiply the result by number 1:", (2 + 5) * 2)
print("Now multiply the result by number 2:", (2 + 5) * 5)
```

```
num1 = 2
num2 = 5

result = num1 + num2
print("The result is:", result)
print("Multiply the result by number 1:", result * num1)
print("Now multiply the result by number 2:", result * num2)
```

A2

If/else statements are conditional statements used to decide on which action should be taken. Without them, there would be no way to choose which path a program should take. In Python, If/else statements are used by saying "if", followed by the condition you want to check, then the code you want to execute if that statement is met. This can either be followed by an "elif", to check another condition if the first one isn't met, or simply an "else" to state the action

that happens if the conditions are not met. In this example, the program will give different outputs depending on if "num1" is equal to 2, or 3, or neither:

```
num1 = 2

if num1 == 2:
    print("Number 1 is equal to 2")
elif num1 == 3:
    print("Number 1 is equal to 3")
else:
    print("Number 1 is not equal to 2 or 3")
```

Multiple conditions can be checked by either using multiple "elif" statements, or by using logical operations such as "and", and "or". These combine multiple checks into the same if statement line. In this example, the program will give different outputs if "num1" is equal to 2 **and** "num2" is equal to 5, or if "num1" is equal to 3 **or** "num2" is equal to 3, or neither:

```
num1 = 2
num2 = 5

if num1 == 2 and num2 == 5:
    print("Number 1 is equal to 2 and number 2 is equal to 5")
elif num1 == 3 or num2 == 3:
    print("Number 1 or number 2 are equal to 3")
else:
    print("Number 1 is not equal to 3, but might be equal to 2")
    print("Number 2 is not equal to 3, but might be equal to 5")
```

A3

Loops are a very important feature of programming, as without them, repeating an action would become drastically less efficient. If you want a specific section of code, or even the whole program, to repeat itself, either a "while" or "for" loop is needed. "for" loops are used to repeat an action for a set number of iterations, then move on when finished. This example will print the value of "num1" (which is 2) 3 times:

```
num1 = 2

for i in range(0, 3, 1):
    print(num1)
```

"while" loops are used to repeat an action until a condition has been met. In this example, the while loop will print the value of "num1" then increase that value by 1, until the value of "num1" is equal to 5:

```
num1 = 2

while num1 < 5:
    print(num1)
    num1 = num1 + 1
```

Part B

Exercise 1 – Data entry

This program allows users to enter their name, age, and height.

Logical Stages

1. Prompt user to enter their name
 - a. If entry type is incorrect, repeat prompt (1)
2. Prompt user to enter their age
 - a. If entry type is incorrect, repeat prompt (2)
3. Prompt user to enter their height
 - a. If entry type is incorrect, repeat prompt (3)
4. Check age is within reasonable range (if not, repeat 2 and 2a *only*)
5. Check height is within reasonable range (if not, repeat 3 and 3a *only*)
6. Print results on screen

Code with annotation

```
def user_name(): # get the user name
    name = str(input("Please enter your name:\n"))
    return name

def user_age(): # get the user age
    check = False # used to ensure valid input
    while not check:
        try:
            age = float(input("Please enter your age:\n"))
        except ValueError:
            print("Please use only numerical values!")
        else:
            check = True
    return age

def user_height(): # get the user height
    check = False # used to ensure valid input
    while not check:
        try:
            height = float(input("Please enter your height in metres:\n"))
        except ValueError:
            print("Please use only numerical values!")
        else:
            check = True
    return height

def main_function():
    userName = user_name() # get the user name
    userAge = user_age() # get the user age
    userHeight = user_height() # get the user height

    while userAge > 120 or userAge < 1: # ensure age values in reasonable
range
        print("There's no way you're that old!")
        userAge = user_age()
```

```
while userHeight > 2.5 or userHeight < 0.5: # ensure height values in
reasonable range
    print("There's no way you're that tall!")
    userHeight = user_height()

    print("Thanks, you entered:", userName, "-", userAge, "years old -",
userHeight, "m tall")

main function()
```

Testing

I. Normal values within range

```
Please enter your name:
James
Please enter your age:
22
Please enter your height in metres:
1.79
Thanks, you entered: James - 22 years old - 1.79 m tall
```

Result: Program performed correctly

II. Boundary

a. Age

```
Please enter your name:
James
Please enter your age:
0
Please enter your height in metres:
1.79
There's no way you're that old!
Please enter your age:
0
There's no way you're that old!
Please enter your age:
22
Thanks, you entered: James - 22 years old - 1.79 m tall
```

Result: Program performed correctly – user was forced to re-enter a value in range

b. Height

```
Please enter your name:
James
Please enter your age:
22
Please enter your height in metres:
0
There's no way you're that tall!
Please enter your height in metres:
2.0
There's no way you're that tall!
Please enter your height in metres:
1.79
Thanks, you entered: James - 22 years old - 1.79 m tall
```

Result: Program performed correctly – user was forced to re-enter a value in range

- III. Test for out-of-range numbers
 - a. Age

```
Please enter your name:
James
Please enter your age:
-10
Please enter your height in metres:
1.79
There's no way you're that old!
Please enter your age:
200
There's no way you're that old!
Please enter your age:
22
Thanks, you entered: James - 22 years old - 1.79 m tall
```

Result: Program performed correctly – user was forced to re-enter a value in range

- b. Height

```
Please enter your name:
James
Please enter your age:
22
Please enter your height in metres:
-10
There's no way you're that tall!
Please enter your height in metres:
299
There's no way you're that tall!
Please enter your height in metres:
1.79
Thanks, you entered: James - 22 years old - 1.79 m tall
```

Result: Program performed correctly – user was forced to re-enter a value in range

IV. Test for incorrect data type
a. Age

```
Please enter your age:
Twenty Two
Please use only numerical values!
Please enter your age:
22
```

Result: Program performed correctly – user was not allowed to enter non-numerical values

b. Height

```
Please enter your age:
22
Please enter your height in metres:
One
Please use only numerical values!
Please enter your height in metres:
1
```

Result: Program performed correctly – user was not allowed to enter a non-numerical value

Exercise 2 – Shape area calculator

This program allows users to calculate the area of a circle, then subtract the area of a square and rectangle from it.

Logical Stages

1. Prompt user to enter circle radius
 - a. If entry type is incorrect or number not above 0, repeat prompt (1)
2. Calculate circle area using formula: $\pi * \text{radius}^2$
3. Prompt user to enter square width
 - a. If entry type is incorrect or number not above 0, repeat prompt (3)
4. Calculate square area using formula: $\text{width} * \text{width}$
5. Prompt user to enter rectangle width
 - a. If entry type is incorrect or number not above 0, repeat prompt (5)
6. Prompt user to enter rectangle length
 - a. If entry type is incorrect or number not above 0, repeat prompt (6)
7. Calculate rectangle area using formula: $\text{width} * \text{length}$
8. Subtract the square and rectangle areas from the circle area
9. If area is a negative, restart program (back to 1)
10. Print results on screen

Code with annotation

```
import math

def circle_function(): # get the circle dimensions
    check = False # used to ensure valid inputs
    while not check or rad <= 0: # ensure radius is in range
        try:
            rad = float(input("Please enter the radius of the circle:\n"))
            if rad <= 0:
                print("Values must be positive numbers and greater than 0")
            except ValueError:
                print("Please only use numerical values!")
            else:
                check = True

    area = math.pi * (rad ** 2) # calculates area of circle
    return area

def square_function(): # get the square dimensions
    check = False # used to ensure valid inputs
    while not check or width < 0: # ensures width is in range
        try:
            width = float(input("Please enter the width of the square:\n"))
            if width < 0:
                print("Values must be positive numbers")
            except ValueError:
                print("Please only use numerical values!")
            else:
                check = True

    area = width * width # calculates area of square
    return area
```

```

def rectangle_function(): # get rectangle dimensions
    check = False # used to ensure valid inputs for width
    while not check or width < 0: # ensures value is within range
        try:
            width = float(input("Please enter the width of the
rectangle:\n"))
            if width < 0:
                print("Values must be positive numbers")
        except ValueError:
            print("Please only use numerical values!")
        else:
            check = True

    check = False # used to ensure valid inputs for length
    while not check or length < 0: # ensures value is within range
        try:
            length = float(input("Please enter the length of the
rectangle:\n"))
            if length < 0:
                print("Values must be positive numbers")
        except ValueError:
            print("Please only use numerical values!")
        else:
            check = True

    area = width * length # calculates area of rectangle
    return area

def shad_area_function(circle, square, rectangle): # calculates shaded
area
    area = circle - square - rectangle
    return area

def main_function():
    circle_area = circle_function() # gets the circle area
    square_area = square_function() # gets the square area
    rectangle_area = rectangle_function() # gets the rectangle area
    shaded_area = shad_area_function(circle_area, square_area,
rectangle_area) # calculates the shaded area

    if shaded_area <= 0: # makes sure that the surface area is larger than
0 - if not, repeat program
        print("The values you have entered will result in a negative shaded
area.")
        print("This is because either the circle is too small, or the other
shapes are too large! ")
        print("Please choose new values!")
        main_function()
    else:
        print("The shaded area is", round(shaded_area, 2), "cm") #
otherwise, print info

main_function()

# tests - ValueErrors, negative numbers, why 0 is not possible only for
circle

```

Testing

I. Normal values within range

```
Please enter the radius of the circle:
10
Please enter the width of the square:
4
Please enter the width of the rectangle:
3
Please enter the length of the rectangle:
5
The shaded area is 283.16 cm
```

Result: Program performed correctly

II. Boundary

a. Circle radius

```
Please enter the radius of the circle:
-1
Values must be positive numbers and greater than 0
Please enter the radius of the circle:
0
Values must be positive numbers and greater than 0
Please enter the radius of the circle:
1
```

Result: Program performed correctly - user was not allowed to enter -1 or 0

b. Square width

```
Please enter the width of the square:
-1
Values must be positive numbers
Please enter the width of the square:
1
```

Result: Program performed correctly - user was not allowed to enter -1

c. Rectangle width

```
Please enter the width of the rectangle:
-1
Values must be positive numbers
Please enter the width of the rectangle:
1
```

Result: Program performed correctly - user was not allowed to enter -1

d. Rectangle length

```
Please enter the length of the rectangle:
-1
Values must be positive numbers
Please enter the length of the rectangle:
1
```

Result: Program performed correctly - user was not allowed to enter -1

III. Test for incorrect data type

a. Circle radius

```
Please enter the radius of the circle:
ten
Please only use numerical values!
Please enter the radius of the circle:
10
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

b. Square width

```
Please enter the width of the square:
four
Please only use numerical values!
Please enter the width of the square:
4
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

c. Rectangle width

```
Please enter the width of the rectangle:
five
Please only use numerical values!
Please enter the width of the rectangle:
5
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

d. Rectangle length

```
Please enter the length of the rectangle:
two
Please only use numerical values!
Please enter the length of the rectangle:
2
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

IV. Test that program won't allow for negative shaded area

```
Please enter the radius of the circle:
2
Please enter the width of the square:
4
Please enter the width of the rectangle:
5
Please enter the length of the rectangle:
10
The values you have entered will result in a negative shaded area.
This is because either the circle is too small, or the other shapes are too large!
Please choose new values!
Please enter the radius of the circle:
```

Result: Program performed correctly – negative shaded value was rejected, and user had to re-enter different values

V. Test for correct calculations

a. Circle area

```
Please enter the radius of the circle:
10
```

$$10 * \pi = 314.16$$

Result: Program performed correctly – calculation correct

b. Square area

```
Please enter the width of the square:
4
```

$$4 * 4 = 16$$

Result: Program performed correctly – calculation correct

c. Rectangle area

```
Please enter the width of the rectangle:
3
Please enter the length of the rectangle:
2
```

$$3 * 2 = 6$$

Result: Program performed correctly – calculation correct

d. Shaded area

```
The shaded area is 292.16 cm
```

$$314.16 - 16 - 6 = 292.16$$

Result: Program performed correctly – calculation correct

Exercise 3 – Petrol cost

This program allows users to enter their mileage for each day of the week, then calculate to total miles travelled, as well as how much that will cost them in petrol.

Logical Stages

1. Prompt user to enter miles for Monday
 - a. If entry type is incorrect or number is negative, repeat prompt (1)
2. Prompt user to enter miles for Tuesday
 - a. If entry type is incorrect or number is negative, repeat prompt (2)
3. Prompt user to enter miles for Wednesday
 - a. If entry type is incorrect or number is negative, repeat prompt (3)
4. Prompt user to enter miles for Thursday
 - a. If entry type is incorrect or number is negative, repeat prompt (4)
5. Prompt user to enter miles for Friday
 - a. If entry type is incorrect or number is negative, repeat prompt (5)
6. Prompt user to enter miles for Saturday
 - a. If entry type is incorrect or number is negative, repeat prompt (6)
7. Prompt user to enter miles for Sunday
 - a. If entry type is incorrect or number is negative, repeat prompt (7)
8. Calculate total miles by adding up each day's miles
9. Print miles on screen
10. Calculate cost using formula: $(\text{miles}/11.2) * 1.35$
11. Print cost on screen

Code with annotation

```
def user_input(): # take the users inputted miles and store them in a list
    if they are valid entries
        day = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday",
               "Saturday", "Sunday"]
        user_miles = []
        position = 0 # the position in the "day" list
        while len(user_miles) < len(day): # repeat until both lists have the
            same number of values
                try: # ensures user enters only float values
                    print("Please enter the miles for", day[position])
                    daily_miles = float(input())
                except ValueError:
                    print("please only use numerical values")
                else:
                    if daily_miles >= 0: # only adds the input to the "user_miles"
                        list and moves to the next position in
                        # "day" list if the input is positive
                        user_miles.append(daily_miles)
                        position = position + 1
                    else:
                        print("Negative miles cannot be entered")
        return user_miles

def total_function(miles): # totals up all the values in the "user_miles"
    array
```

```

total = 0
for i in range(0, len(miles)):
    total = total + miles[i]
return total

def cost_function(miles): # calculates the cost of petrol
    cost = (miles / 11.2) * 1.35 # 11.2 miles to the litre, £1.35 per
    litre
    return cost

def main_function(): # main function
    dailyMiles = user input()
    totalMiles = total_function(dailyMiles)
    totalCost = cost_function(totalMiles)
    print(totalMiles, "total miles")
    print("Cost of petrol:", round(totalCost, 2))

main_function()

```

Testing

I. Normal values within range

```

Please enter the miles for Monday
2
Please enter the miles for Tuesday
3
Please enter the miles for Wednesday
4
Please enter the miles for Thursday
5
Please enter the miles for Friday
6
Please enter the miles for Saturday
7
Please enter the miles for Sunday
8
35.0 total miles
Cost of petrol: 4.22

```

Result: Program performed correctly

II. Boundary

a. Monday

```
Please enter the miles for Monday
-1
Negative miles cannot be entered
Please enter the miles for Monday
1
```

Result: Program performed correctly - user was not allowed to enter -1

b. Tuesday

```
Please enter the miles for Tuesday
-1
Negative miles cannot be entered
Please enter the miles for Tuesday
1
```

Result: Program performed correctly - user was not allowed to enter -1

c. Wednesday

```
Please enter the miles for Wednesday
-1
Negative miles cannot be entered
Please enter the miles for Wednesday
1
```

Result: Program performed correctly - user was not allowed to enter -1

d. Thursday

```
Please enter the miles for Thursday
-1
Negative miles cannot be entered
Please enter the miles for Thursday
1
```

Result: Program performed correctly - user was not allowed to enter -1

e. Friday

```
Please enter the miles for Friday
-1
Negative miles cannot be entered
Please enter the miles for Friday
1
```

Result: Program performed correctly - user was not allowed to enter -1

f. Saturday

```
Please enter the miles for Saturday
-1
Negative miles cannot be entered
Please enter the miles for Saturday
1
```

Result: Program performed correctly - user was not allowed to enter -1

g. Sunday

```
Please enter the miles for Sunday
-1
Negative miles cannot be entered
Please enter the miles for Sunday
1
```

Result: Program performed correctly - user was not allowed to enter -1

III. Test for incorrect data type

a. Monday

```
Please enter the miles for Monday
one
please only use numerical values
Please enter the miles for Monday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

b. Tuesday

```
Please enter the miles for Tuesday
one
please only use numerical values
Please enter the miles for Tuesday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

c. Wednesday

```
Please enter the miles for Wednesday
one
please only use numerical values
Please enter the miles for Wednesday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

d. Thursday

```
Please enter the miles for Thursday
one
please only use numerical values
Please enter the miles for Thursday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

e. Friday

```
Please enter the miles for Friday
one
please only use numerical values
Please enter the miles for Friday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

f. Saturday

```
Please enter the miles for Saturday
one
please only use numerical values
Please enter the miles for Saturday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

g. Sunday

```
Please enter the miles for Sunday
one
please only use numerical values
Please enter the miles for Sunday
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

IV. Test for correct calculations

a. Total miles

```
Please enter the miles for Monday
1
Please enter the miles for Tuesday
2
Please enter the miles for Wednesday
3
Please enter the miles for Thursday
4
Please enter the miles for Friday
5
Please enter the miles for Saturday
6
Please enter the miles for Sunday
7
28.0 total miles
```

$1 + 2 + 3 + 4 + 5 + 6 + 7 = 28$

Result: Program performed correctly – calculation correct

b. Cost of petrol

```
28.0 total miles
Cost of petrol: 3.38
```

$(28 / 11.2) * 1.35 = 3.38$ (11.2 miles to the litre, £1.35 per litre)

Result: Program performed correctly – calculation correct

Exercise 4 – Pairs game

This program is a game in which users have to correctly match pairs of fruits together.

Logical Stages

1. Randomly assign positions to the list of pairs
2. Print numbers 0 to 11 on screen
3. Print list containing 12 "?"s on screen
4. If this is not the user's first turn skip, tell them how many turns they have had (if it is the first turn, skip to 6)
5. If all of the pairs have been found, congratulate the user, and tell them how many turns it took (if not, skip to 6)
6. Prompt user to enter first position
 - a. If entry type is incorrect, number is not a whole number between 0 and 11, or position has already been marked as "X", repeat prompt (6)
7. Prompt user to enter second position
 - a. If entry type is incorrect, number is not a whole number between 0 and 11, or position has already been marked as "X", repeat prompt (7)
8. If the chosen positions are a matching pair, tell the user, then replace the corresponding positions in the "?" list with "X"s (back to 2)
9. If the chosen positions are not a matching pair, tell the user what each position is (back to 2)

Code with annotation

```
import random

def print_function(hidden): # function to print out the line of positions
and the line of "?" or "X"s
    for i in range(0, 12):
        print(i, end=" ")
    print()
    for i in hidden:
        print(i, end=" ")
    print()

def user_choice_function(hidden): # takes user input and ensures that it
has not already been picked
    # print(answerList) # for testing the position in the list
    ***** TESTING COMMAND *****
    check1 = False # used for ensure int input
    while not check1:
        try:
            choice = int(input())
        except ValueError:
            print("Please only enter whole, positive, numerical values")
        else:
```

```

        if choice < 0 or choice > 11:
            print("Please only choose numbers from the list")
        else:
            check1 = True

    check2 = False # used to force user to pick a different position
    while not check2:
        if hidden[choice] == "X":
            print("That position has already been revealed, Please choose
another:")
            choice = int(input())
        else:
            if choice < 0 or choice > 11:
                print("Please only choose numbers from the list")
            else:
                check2 = True
    return choice

def correct_function(iteration, answer, hidden, choice1, choice2):
    if answer[choice1] == answer[choice2]: # if the 2 user choices match,
change the "?" in the hidden list to an "x",
    # and tell the user they are correct
    print("Correct! You have matched the", answer[choice1], "pair")
    hidden[choice1] = "X"
    hidden[choice2] = "X"
    print_function(hidden) # reprint the lists for the user
    iteration = iteration - 1 # mark that a pair has been found
    print()
    else: # tells user their guess is incorrect
    print("Incorrect")
    print("Position", choice1, "is", answer[choice1], "and Position",
choice2, "is",
        answer[choice2])
    print()
    print_function(hidden) # reprint the lists for the user
    print()
    return iteration # will set that a pair has been found

def list_function():
    hidden_list = ["?", "?", "?", "?", "?", "?", "?", "?", "?", "?", "?", "?",
"?"] # stores the "?" and "X" for checking
    # pairs
    sample_list = ["Apple", "Pear", "Orange", "Banana", "Grape",
"Blueberry"] # stores the possible pairs
    answer_list = [] # stores the positions of the pairs

    for i in range(0, 2): # stores 2 copies of each item from sample_list
in the answer list, distributed randomly
        for j in range(0, len(sample_list)):
            answer_list.insert(random.randint(0, len(answer_list)),
sample_list[j])

    combo_list = [hidden_list, answer_list] # stores both lists in a list
so that the values can be retrieved and
    # separated again outside the function
    return combo_list

# ***** Start of program *****

```

```

complete = 6 # total number of pairs to be selected, so that the game
knows when to end
turnCounter = 0 # total number of turns the user has taken

comboList = list_function() # retrieve the hidden and answerList from the
function
hiddenList = comboList[0] # retrieves the hiddenList from the comboList
answerList = comboList[1] # retrieves the answerList from the comboList

print_function(hiddenList) # print the "?" mark list

while complete > 0: # repeat the program whilst there are still pairs
remaining

    print("Pick first position: ")
    userChoice1 = user_choice_function(hiddenList) # get the users first
choice
    print("Pick second position: ")
    userChoice2 = user_choice_function(hiddenList) # get the users second
choice
    print()
    if userChoice1 == userChoice2: # prevent the user from picking the
same letter for choice 1 and 2
        print("You can't pick the same position!")
    else:
        complete = correct_function(complete, answerList, hiddenList,
userChoice1, userChoice2) # checks if the users
# guess was correct
        turnCounter = turnCounter + 1 # increase the turn counter
        print(f"You've had {turnCounter} turn(s) so far!") # tell the
user how many turns they've had

print(f"Congratulations, you got them all! You did it in {turnCounter}
turns") # tell the user they have won and how
# many turns it took them

```

Testing

I. Normal values within range

```

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ?
Pick first position:
1
Pick second position:
4

```

Result: Program performed correctly

II. Boundary

a. Position 1

```
Pick first position:  
-1  
Please only choose numbers from the list  
12  
Please only choose numbers from the list
```

Result: Program performed correctly - user was not allowed to enter -1 or 12

b. Position 2

```
Pick second position:  
-1  
Please only choose numbers from the list  
12  
Please only choose numbers from the list
```

Result: Program performed correctly - user was not allowed to enter -1 or 12

III. Test for incorrect data type

a. Position 1

```
Pick first position:  
one  
Please only enter whole, positive, numerical values  
1
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

b. Position 2

```
Pick second position:  
zero  
Please only enter whole, positive, numerical values  
0
```

Result: Program performed correctly - user was not allowed to enter a non-numerical value

IV. Test for entering the same position

```

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ?
Pick first position:
0
Pick second position:
0

You can't pick the same position!

```

Result: Program performed correctly - user was not allowed to enter the same number in both positions

- V. Test for entering position already marked as a found pair
- Position 1

```

0 1 2 3 4 5 6 7 8 9 10 11
? X ? X ? ? ? ? ? ? ?

You've had 3 turn(s) so far!
Pick first position:
1
That position has already been revealed, Please choose another:

```

Result: Program performed correctly – program recognised the position was already found and user was forced to pick a different one

- Position 2

```

Pick second position:
1
That position has already been revealed, Please choose another:

```

Result: Program performed correctly – program recognised the position was already found and user was forced to pick a different one

- VI. Test for pair randomiser

Game 1

```

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ?
Pick first position:
0
Pick second position:
1

Incorrect
Position 0 is Blueberry and Position 1 is Grape

```

Game 2

```

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?
Pick first position:
0
Pick second position:
1

Incorrect
Position 0 is Banana and Position 1 is Apple

```

Result: Program performed correctly – positions 0 and 1 had different pairs in different games

VII. Test for turn counter increase

```

Incorrect
Position 0 is Grape and Position 1 is Blueberry

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?

You've had 1 turn(s) so far!
Pick first position:
2
Pick second position:
3

Incorrect
Position 2 is Pear and Position 3 is Orange

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?

You've had 2 turn(s) so far!

```

Result: Program performed correctly – turn counter increases after the user makes an in/correct guess

VIII. Test for correct guess

```

Pick first position:
3
Pick second position:
4

Correct! You have matched the Orange pair
0 1 2 3 4 5 6 7 8 9 10 11
? ? ? X X ? ? ? ? ? ? ?

```

Result: Program performed correctly – matching pair's "?"s were replaced with "X"s

IX. Test for incorrect guess

```
0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?
Pick first position:
0
Pick second position:
1

Incorrect
Position 0 is Orange and Position 1 is Banana

0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?
```

Result: Program performed correctly – non-matching pair's "?"s were not replaced with "X"s

X. Test for completing game

```
Correct! You have matched the Grape pair
0 1 2 3 4 5 6 7 8 9 10 11
X X X X X X X X X X X X

You've had 11 turn(s) so far!
Congratulations, you got them all! You did it in 11 turns
```

Result: Program performed correctly – game ends when all pairs have been found and prints how many turns it took the user

Software Development Assignment James Frew 2108660

GRADEMARK REPORT

FINAL GRADE

95 / 100

GENERAL COMMENTS

Instructor

PART A

Variables - You have talked about variables and types, and inferring types. You could also mentioned that in Python, types can change within the lifetime of the program, (for example, when dividing two integers, the resulting type is changed to float) 4 /5

If/Else - You've got the main points here with suitable examples. 5/5

Loops - You've got the main points with suitable examples. 5/5

PART B

Exercise 1

Logical Stages-Testing - Logic is OK, but you need to state what "reasonable range" is so the programmer is able to implement it. Testing is good 4.5 /5

Coding-Annotation - Good 10 /10

Exercise 2

Logical Stages-Testing - Good 5/5

Coding-Annotation - Good program, you consider size of circle against the square/rectangle, could also consider size of square/rectangle against each other. 9/10

Exercise 3

Logical Stages-Testing - Good, and thorough testing presented 10/10

Coding-Annotation - Good, could also have validated an upper limit of miles as well as lower. 9/10

Exercise 4

Logical Stages-Testing - Generally clear on both counts. 9 /10

Coding-Annotation - A good effort. You might want to look at `.shuffle()` to simplify the randomisation of the list order, but overall good use of functions and validation is thorough. Well done. 24 /25

General comment: All assignments should really have a short Introduction and a conclusion - just to set the scene on what the assignment is about and in the conclusion, to discuss how things went and what you would do differently next time. There are no marks assigned for that in this particular assignment, but it is good academic practice.

Assignment overall: 94.5%

Test mark: 89%

OVERALL MARK FOR MODULE: 93%

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

PAGE 14

PAGE 15

PAGE 16

PAGE 17

PAGE 18

PAGE 19

PAGE 20

PAGE 21
