

Student: Dan Tataru

Student ID: 2125053

Course: Introduction to Web and Database Concepts

Module code: ACCB4017

Assessment 1

A1. What is a variable and why are they important in writing programs? Discuss the way variables are handled in the programming language you have studied (eg static vs dynamic typing)

The names you provide to computer memory regions used to hold data in computer programmes are known as variables. Assume, for instance, that you wish to save the values 10 and 20 in your application and then use them later. Let's see what you're going to do. Here are the next three easy steps: Make up variables with sensible names. These two variables should have your values.

The variables' stored values can be retrieved and used. A variable name may include only one kind of value. For instance, if the variable `a` has an int type definition, it can only hold int values.

Before using a variable in your programme, you must declare it in the C programming language. Although a programming language like Python permits you to use a variable name without generating it, you cannot use a variable name in your programme without first creating it.

A variable name may only be used once in your application. You cannot declare a variable again to hold any other sort of value, for instance, if it has already been defined to store an integer value. Some programming languages, like Python, PHP, Perl, and others, do not need you to specify the data type when creating variables. Thus, you don't need to declare the data type of an integer, float, or long when storing them.

A2. Explain what an if/else statement is, how and why they are used, and give a code example of when you would use one. Include in your answer a discussion of the use of multiple conditions.

Throughout the many programming languages, conditional statements are used to advise the computer on the choice to be made when certain criteria are met. Depending on the functions the programmer has in mind, these selections are only taken if and only if the previously mentioned conditions are either true or untrue.

Example

```
x = 100
```

```
y = 20
```

```
if x < y:
```

```
    print("x is lesser than y")
```

```
elif x == y:
    print("x and y are same")
else:
    print("x is greater than y")
```

While the if statement may be used to verify one condition, else if is used to examine many criteria. The else if block is followed by an expression with syntax identical to the if statement (or elif in Python). The criteria are evaluated top-down in else if statements; if the first block returns true, the second and third blocks are not examined; but, if the first if block returns false, the second block is. This checking keeps on until a block yields a true result.

A3. Explain why loops are needed in a program and when you might use a for and a while loop. Give suitable examples of each in the programming language you have studied.

You can get across circumstances where a block of code has to be run several times. The first statement in a function is executed first, then the second, and so on. In general, statements are executed in order. Different control structures offered by programming languages enable more intricate execution routes. We may run a statement or set of statements repeatedly by using a loop statement. While loops are used to repeatedly run a block of code up until a predetermined condition is satisfied. Let's take an example where we want to know how many times we can divide a certain integer by two until the result is less than or equal to one.

While loop in C

```
while (i <= 5) {
    printf("%d\n", i);
    ++i;
}
```

While loop in Python

```
while True:
```

```
print(i)
i = i + 1
if(i > 3):
    break
```

Exercise 01

Solution:

Name, age, and height are the three inputs for this activity. Age, height, and name are all strings. Height is a float. There must be a non-numeric character in the name. Both a height range of less than 2.5 metres and an age range of 0 to 100 must be satisfied. The age and height string values should now be converted to integers and floats, respectively. Print the final outcome after merging the last three inputs.

Code:

```
# loop until user doesn't enter valid values
while True:
    # take input from user as a name
    user_name = input("Please enter your name:\n")
    if user_name.isalpha():
        break
    else:
        print("Unless, you are from mars, that seems too odd!")

# loop until user doesn't enter valid values
while True:
```

```
# take input from the user as an age
user_age = int(input("Please enter your age:\n"))
if user_age < 0:
    print("Unless, you are not born yet!")
elif user_age > 100:
    print(
        "Unless, you are from marvel university, that seems you are living very much!")
else:
    break
```

```
# loop until user doesn't enter valid values
while True:
    # take input from the user as a height
    user_height = float(input("Please enter your height:\n"))
    if user_height > 2.5:
        print("Unless, you are a giant, that seems a big high!")
    else:
        break
```

```
# output
print(
    f"Thanks, you entered {user_name} - {user_age} years old - {user_height} cm tall")
```

Testing:

Test 1

```
Please enter your name:
David123
Unless, you are from mars, that seems too odd!
Please enter your name:
█
```

The user must specify a name without any numeric characters to prevent error notifications. Otherwise, unless the supplied name is suitable, it will ask for the user name again.

Test 2

```
Please enter your name:
David
Please enter your age:
-5
Unless, you are not born yet!
Please enter your age:
█
```

The user must specify an age that ranges from 0 to 100 in order to prevent receiving an error notice. If the provided age is invalid, it will ask for it again.

Test 3

```
Please enter your name:
David
Please enter your age:
-5
Unless, you are not born yet!
Please enter your age:
20
Please enter your height:
4.7
Unless, you are a giant, that seems a big high!
Please enter your height:
█
```

The user must provide a height that is lower than 2.5 metres in order to prevent receiving an error notice. Otherwise, unless the height provided is appropriate, it will ask for the height again.

Test 4

```
Please enter your name:
David
Please enter your age:
20
Please enter your height:
1.7
Thanks, you entered David - 20 years old - 1.7 cm tall
```

The user will receive a successful outcome if they enter three valid inputs.

Exercise 02

Solution:

For this activity, the user will need to provide four inputs: radius, side, length, and width. All four inputs are float data types. The circle's radius is determined using the radius variable. Rectangles and squares are measured using the length and width variables, respectively. To determine the areas of a circle, square, and rectangle, three formulae are utilised.

Code:

```
# loop until user doesn't enter valid radius of the circle
while True:
    # take input from user as radius
    radius_of_circle = input("Please enter radius of circle:\n")
    if radius_of_circle.isdigit():
        radius_of_circle = float(radius_of_circle)
        break
    else:
        print("That seems to be invalid radius, please try again:")

# loop until user doesn't enter valid side of the square
while True:
    # take input from user as side
    side_of_square = float(input("Please enter side of square:\n"))
    if side_of_square > radius_of_circle*2:
        print("That seems a little big, please try again:")
    else:
        break

# loop until user doesn't enter valid width of the rectangle
while True:
    # take input from user as width
    width_of_rectangle = float(input("Please enter width of rectangle:\n"))
```

```
if width_of_rectangle > radius_of_circle*2:
    print("That seems a little big, please try again:")
else:
    break
```

```
# loop until user doesn't enter valid length of the rectangle
while True:
    # take input from user as length
    length_of_rectangle = float(input("Please enter length of rectangle:\n"))
    if length_of_rectangle > radius_of_circle*2:
        print("That seems a little big, please try again:")
    else:
        break
```

```
total_area_of_circle = 3.14 * \
    (radius_of_circle ** 2) # calculated total area of circle
total_area_of_square = side_of_square ** 2 # calculated total area of square
total_area_of_rectangle = width_of_rectangle * \
    length_of_rectangle # calculated total area of rectangle

# calculated blue shaded area of the circle
shaded_area = total_area_of_circle - \
    total_area_of_square - total_area_of_rectangle

# print the result
print(f"The shaded area is {shaded_area} cm")
```

Testing:

Test 1

```
Please enter radius of circle:
asdas
That seems to be invalid radius, please try again:
Please enter radius of circle:
█
```

The user is required to input a number value in order to prevent error warnings. Otherwise, unless the provided radius is not a number, it will ask for the circle's radius once again.

Test 2

```
Please enter radius of circle:
7
Please enter side of square:
10
Please enter width of rectangle:
40
That seems a little big, please try again:
Please enter width of rectangle:
█
```

The user must specify a proper width and length for the rectangle that will be included within the circle in order to prevent receiving an error notice. Otherwise, unless the specified width and length are turned into a rectangle within a circle, it will ask for the width or length of the rectangle once more.

Test 3

```
Please enter radius of circle:
5
Please enter side of square:
2
Please enter width of rectangle:
12
That seems a little big, please try again:
Please enter width of rectangle:
2
Please enter length of rectangle:
1
The shaded area is 72.5 cm
```

A successful outcome and the darkened area of the circle in centimetres will be returned if the user provides four correct inputs.

Exercise 03

Solution:

Two lists were required for this exercise: one static list for the names of the days of the week and one dynamic list for the number of kilometres driven on each day.

When calculating the miles of a day in a list, certain aspects are taken into account, such as whether the list exclusively contains numeric numbers and if it contains larger than 0 values.

The method may then be used to determine the total number of miles driven by the automobile over the week as well as the cost of gas.

Code:

```
# name of days
name_of_days = ["Monday", "Tuesday", "Wednesday",
                "Thursday", "Friday", "Saturday", "Sunday"]
miles_of_car = [] # empty list

# for loop for day in each week
for day in name_of_days:
    # loop until user doesn't enter valid range of the miles
    while True:
        miles_for_day = input(f'Enter miles for {day} :\n')
        if miles_for_day.isalpha() or float(miles_for_day) < 0:
            print("Please try again")
        else:
            miles_for_day = float(miles_for_day)
            miles_of_car.append(miles_for_day)
            break

total_miles = sum(miles_of_car) # calculate total miles
# calculate cost of petrol of total miles
cost_of_petrol = round((total_miles/11.2)*1.35, 2)
```

```
# print the results
```

```
print(f"The total mileage was: {total_miles}")
```

```
print(f"The cost of petrol : {cost_of_petrol}")
```

Testing:

Test 1

```
Enter miles for Monday :  
10  
Enter miles for Tuesday :  
-2  
Please try again  
Enter miles for Tuesday :  
□
```

The user must input a numeric value with a positive range, or larger than 0, in order to avoid error warnings. Unless the provided vehicle mile is more than 0, it will then ask for the car's mile again.

Test 2

```
Enter miles for Monday :  
3  
Enter miles for Tuesday :  
4  
Enter miles for Wednesday :  
50  
Enter miles for Thursday :  
23  
Enter miles for Friday :  
62  
Enter miles for Saturday :  
92  
Enter miles for Sunday :  
12  
The total mileage was: 246.0  
The cost of petrol : 29.65
```

The final successful result will show the total distance and the cost of gasoline for the week if the user inputs legitimate miles for each day.

Exercise 04

In this exercise, a list with 12 question mark values is formed, a static key-value pair is constructed for each pair of the board game, and an empty list is created to fill in the spaces on the board that are not occupied. The functions display board, play game, and is game over are developed. Using display board, the current board is shown. and X signs. The ? The sign denotes a location that is open, whereas the X denotes a location that is closed. The play game method takes two arguments, choice 1 and choice 2, which are used to check if a pair of options is present or not. A success message is shown and options are added to the not available position if a pair is found. Check if the game is over using the is game over function.

Code:

```
game_board = ['?', '?', '?', '?', '?', '?', '?', '?', '?',
              '?', '?', '?'] # a game board
game_pairs = {0: 'dog', 1: 'fish', 2: 'duck', 3: 'lion', 4: 'elephant', 5: 'mouse',
              6: 'dog', 7: 'fish', 8: 'duck', 9: 'lion', 10: 'elephant', 11: 'mouse'} # pairs
untouchable_positions = [] # empty list

# show the board
def show_board():
    for i in range(0, len(game_board)):
        print(i, end=" ")

    print()
    for place in game_board:
        print(place, end=" ")

    print()
```

```

# start a game
def start_game(choice_01, choice_02):
    found_pair = False
    # check for pair
    if game_pairs.get(choice_01) == game_pairs.get(choice_02):
        game_board[choice_01] = 'X'
        game_board[choice_02] = 'X'
        found_pair = True

    # display message according to pair matched or not
    if not found_pair:
        print(
            f'Not a match. Position {choice_01} was {game_pairs.get(choice_01)} and position
{choice_02} was {game_pairs.get(choice_02)}')
    else:
        print(f'Well done, that's a match on {game_pairs.get(choice_01)} !')
        untouchable_positions.append(choice_01)
        untouchable_positions.append(choice_02)
        show_board()

if __name__ == '__main__':
    print("Welcome to the pairs game! There are 6 pairs to find")
    print("Here is the board:")
    show_board() # show the board
    count_for_turns = 0 # count the number of turns

    # loop until board is full of X with pairs
    while True:
        # loop until player choose valid board place
        while True:

```

```

choice_01 = int(input("\nPlease enter guess 1:"))
# check place is available
if choice_01 in untouchable_positions:
    print("That one is not available! Please try again\n")
else:
    break

# loop until player choose valid board place
while True:
    choice_02 = int(input("Please enter guess 2:"))
    # check place is available
    if choice_02 in untouchable_positions:
        print("That one is not available! Please try again\n")
    else:
        break

start_game(choice_01, choice_02) # start the game again
# count the number of turns the player has played
count_for_turns = count_for_turns+1
print(f'You've had {count_for_turns} turn(s) so far')

# check if the game is over
if not '?' in game_board:
    break

# print the final result
print(
    f'Well done, you got them all. you did it in {count_for_turns} turns')

```

Testing:

Test 1

```
Welcome to the pairs game! There are 6 pairs to find
Here is the board:
0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?

Please enter guess 1:5
Please enter guess 2:6
Not a match. Position 5 was mouse and position 6 was dog
You've had 1 turn(s) so far

Please enter guess 1:█
```

The user must input the correct estimation of pair in numeric value to avoid error notifications.

If not, it will indicate that the pairs are mismatched and increase the number of rotations.

Test 2

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

Please enter guess 1:5
Please enter guess 2:6
Not a match. Position 5 was mouse and position 6 was dog
You've had 1 turn(s) so far

Please enter guess 1:1
Please enter guess 2:7
Well done, that's a match on fish !
0 1 2 3 4 5 6 7 8 9 10 11
? X ? ? ? ? ? X ? ? ? ?
You've had 2 turn(s) so far

Please enter guess 1:█
```

If you correctly guess the pair, you'll receive a message of success and a turn? sign to X symbol on the board, indicating the matched pair.

Test 3

```
Welcome to the pairs game! There are 6 pairs to find
Here is the board:
0 1 2 3 4 5 6 7 8 9 10 11
? ? ? ? ? ? ? ? ? ? ? ?

Please enter guess 1:5
Please enter guess 2:6
Not a match. Position 5 was mouse and position 6 was dog
You've had 1 turn(s) so far

Please enter guess 1:1
Please enter guess 2:7
Well done, that's a match on fish !
0 1 2 3 4 5 6 7 8 9 10 11
? X ? ? ? ? ? X ? ? ? ?
You've had 2 turn(s) so far

Please enter guess 1:2
Please enter guess 2:1
That one is not available! Please try again

Please enter guess 2: 
```

The user must enter the correct answer at the appropriate location that will be shown on the board in order to avoid receiving an error notice. If not, it will prompt you to guess again unless you enter a space on the board that is open and not marked with an X.

Test 4

```
Please enter guess 1:4
Please enter guess 2:10
Well done, that's a match on elephant !
0 1 2 3 4 5 6 7 8 9 10 11
X X X X X ? X X X X X ?
You've had 7 turn(s) so far

Please enter guess 1:5
Please enter guess 2:11
Well done, that's a match on mouse !
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 8 turn(s) so far
Well done, you got them all. you did it in 8 turns
```

At the conclusion of the game, the user will get a positive outcome if they correctly predicted the board's pair values. Additionally, they will learn how many turns it took them to find the correct pairs.