

Software Dev Assignment 1

by Jonty Phillips

Submission date: 26-Apr-2022 07:13PM (UTC+0100)

Submission ID: 177387476

File name: 36119_Jonty_Phillips_Software_Dev_Assignment_1_410175_1801021399.docx (1.6M)

Word count: 2534

Character count: 12523

Software Dev Assignment 1

Python

PART A (UNDERSTANDING)

- 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)*

A variable in python is important for writing programs because it allows the programmer to assign values or tags to a word. This is incredibly useful as they act as a form of storage, for example a variable could be as simple as "inputOne" for a calculator – that stores a number, or "userProfile" for the user's name.

```
myName = "Jonty"  
myNumber = 7  
myPrice = 3.141
```

Variables can be laid out in a number of different ways, one of the most common (and my most comfortable) is nicknamed "camelCase" and is written as the first word all lowercase and the following words start with an uppercase. This is used as spaces are invalid in naming variables, (the only special character allowed being an underscore) and it keeps variables concise to not use much space.

Python has the added bonus of being a dynamically typed language, as the variable's "type" is automatically deduced during the program's runtime. This is helpful as it allows the programmer to simply not worry about any specificity with variables if they so wish, however many bugs/issues could also arise from the program auto assigning the type (e.g. labelling a string as a float or vice versa)

- 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.*

An if/else statement functions as a sort of filter in a program, essentially by predicting one of two outcomes and assigning the pathway based on parameters set by the programmer. They are used by stating: (square brackets used for clarity, would not be used in code) -[if (then your parameter, eg a variable name) == (a certain result).], then on a new indented line the programmer writes the code that will run as a result of that variable input. Underneath this [if] code, the programmer would also state [else:] which encompasses all other cases that don't come under the specified parameter (in this case any input variable that isn't the specified one).

```
myName = input()  
  
if myName == "Jonty":  
    print("that's me :)")  
else:  
    print("that's not me :(")
```

If for any reason the programmer needs more than one specific outcome to filter, *[elif]* can be used multiple times the same as an if statement, for example:

```
userYN = input("yes or no?")

if userYN == "yes":
    print("Yes Sir!")

elif userYN == "no":
    print("No Problem!")

else:
    print("that's an option :(")
```

In this example I have encompassed all options I believe the user to choose given the question asked, however a programmer should always have a route for if a user doesn't do as predicted to prevent an exit code.

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

Loops are an essential part of programming as they help minimise the code/potential errors in repeating lines of code for repetitive tasks, of which are very common in programming. For loops are defined when the number of potential iterations is known by the programmer or computer, useful for checking attendance in a class list database for example.

```
for myCounter in range(0, 13, 1):
    myTotal = myCounter * 7
    print(myCounter, 'times 7 is', myTotal)

print('end')
```

While loops however are for repeating conditions that are of an unknown amount until a specific parameter is hit e.g., a spending budget variable in a shopping calculator.

```
userBudget = float(500.0)

while userBudget != 0 or userBudget < 0:
    userSpent = float(input("how much did you spend?"))
    userBudget = userBudget - userSpent
    print(userBudget)

print("Budget spent!")
```

PART B (IMPLEMENTATION)

Exercise 1:

Show how variables can contain different types of values by declaring three variables that will contain a name, a measure of height (e.g. 1.84m) and a person's age (e.g. 26). Prompt the user to input these values and display them on the screen. The variables should be appropriately typed for the data required.

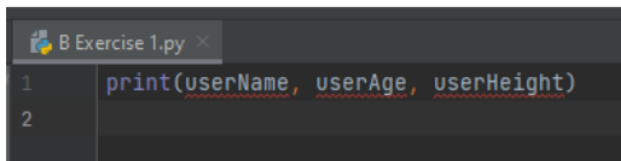
```
# program to display user's name, age and height

userName = str(input('Please state your name: '))    # user's name
userAge = int(input('Please enter your age: '))      # user's age
userHeight = float(input('Please enter your height (in metres): '))    # user's height

while userAge == 0:                                # this is to prevent user from inserting an age too low
    userAge = int(input('Your age cannot be 0! Try again:'))

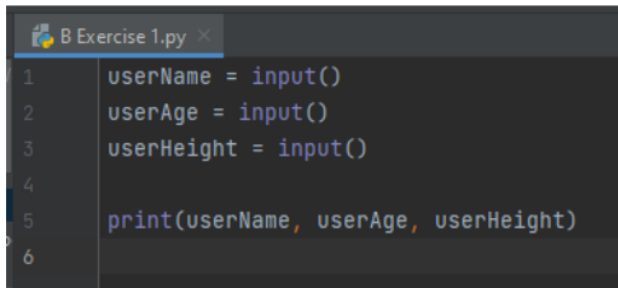
while userHeight >= 3:                              # this is to prevent the user from inserting a height that
    is too large
    userHeight = int(input('Is this correct? That's very high! Input again: '))

print(userName, ' - ', userAge, ' years old - ', userHeight, 'm tall', sep="")
# this is to neatly display the users results
```



```
B Exercise 1.py x
1 print(userName, userAge, userHeight)
2
```

The author started off by stating the variables in the desired order of the end print.



```
B Exercise 1.py x
1 userName = input()
2 userAge = input()
3 userHeight = input()
4
5 print(userName, userAge, userHeight)
6
```

Next they defined the variables in the same order, preparing to add the base print text for the user.

```
1 userName = str(input('Please state your name: '))
2 userAge = int(input('Please enter your age: '))
3 userHeight = float(input('Please enter your height (in metres): '))
4
5 print(userName, userAge, userHeight)
6
```

They also specified the type of variable to prevent any confusion.

The screenshot shows the VS Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows a project named 'Gaynor' with a subfolder 'Assignment 1 - 29th April' containing '8 Exercise 1.py'. The editor shows the code from the previous block. The terminal shows the output of running the code:

```
"C:\Users\jonty\OneDrive\Documents\1. Deep Docs\1. Work\Uni\Gaynor\venv\Scripts\python.exe" "C:/Users/jonty/OneDrive/
Please state your name: test
Please enter your age: 1
Please enter your height (in metres): 1.0
test - 1 - 1.0
Process finished with exit code 0
```

At first they thought this would work, until they coded the values names (yrs old, m).

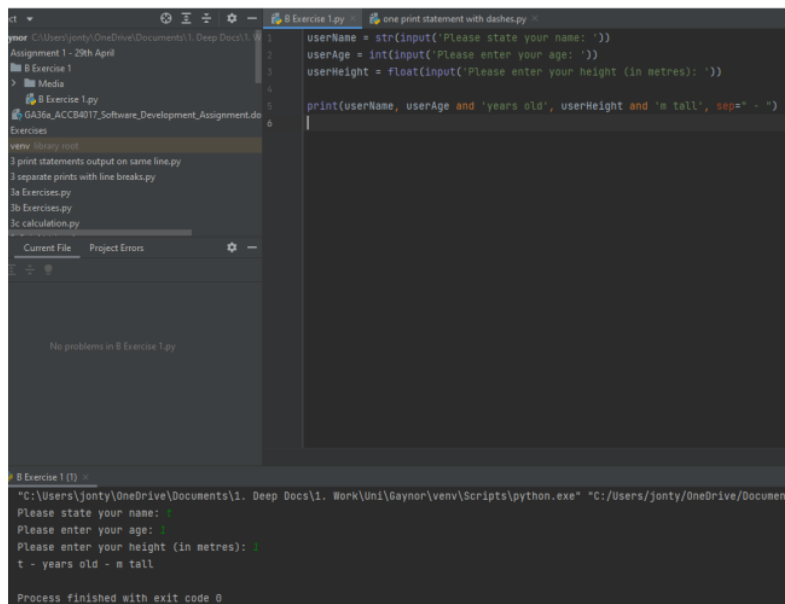
The screenshot shows the VS Code interface with the file explorer on the left, the editor in the center, and the terminal at the bottom. The file explorer shows the same project structure. The editor shows the modified code:

```
1 userName = str(input('Please state your name: '))
2 userAge = int(input('Please enter your age: '))
3 userHeight = float(input('Please enter your height (in metres): '))
4
5 print(userName, userAge, 'years old', userHeight, 'm tall', sep=" - ")
6
```

The terminal shows the output of running the modified code:

```
"C:\Users\jonty\OneDrive\Documents\1. Deep Docs\1. Work\Uni\Gaynor\venv\Scripts\python.exe" "C:/Users/jonty/OneDrive/D
Please state your name: test
Please enter your age: 1
Please enter your height (in metres): 1.0
test - 1 - years old - 1.0 - m tall
Process finished with exit code 0
```

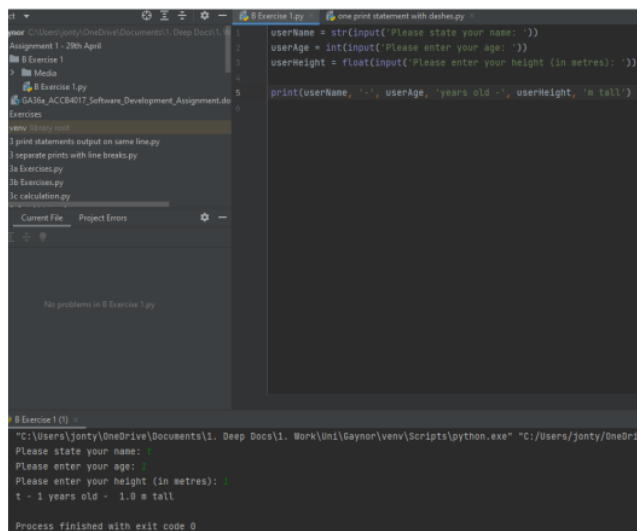
This gave them a bit of grief to solve.



```
1 userName = str(input('Please state your name: '))
2 userAge = int(input('Please enter your age: '))
3 userHeight = float(input('Please enter your height (in metres): '))
4
5 print(userName, userAge and 'years old', userHeight and 'm tall', sep=" - ")
6
```

Process finished with exit code 0

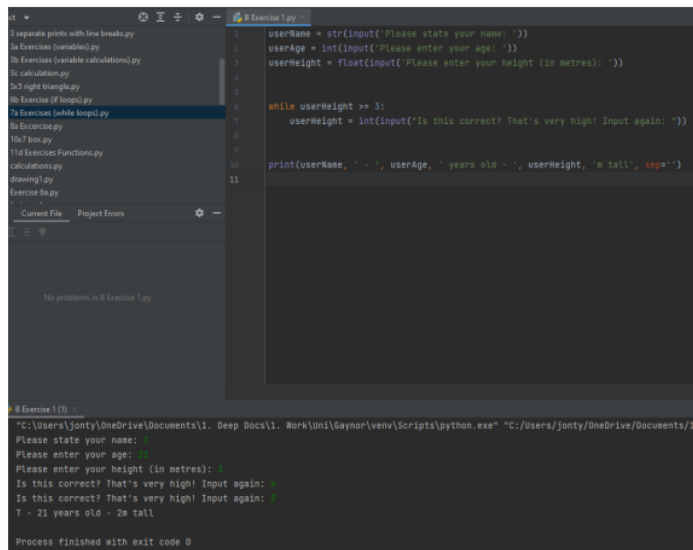
This was an attempted solution for the problem however this caused the values to not be shown.



```
1 userName = str(input('Please state your name: '))
2 userAge = int(input('Please enter your age: '))
3 userHeight = float(input('Please enter your height (in metres): '))
4
5 print(userName, '-', userAge, 'years old -', userHeight, 'm tall')
6
```

Process finished with exit code 0

Finally, a solution was found that was roughly what they wanted, the result did bother them slightly with the unnecessary gaps included but, they were happy.



```
1 separate prints with line breaks.py
2 Exercises (variables).py
3 Exercises (variable calculations).py
4 calculations.py
5 5's right triangle.py
6 Exercise (if loops).py
7 Exercise (while loops).py
8 Exercises.py
9 10's flow.py
10 Exercises Functions.py
11 calculations.py
12 drawing1.py
13 Exercise 8a.py

Current File Project Errors

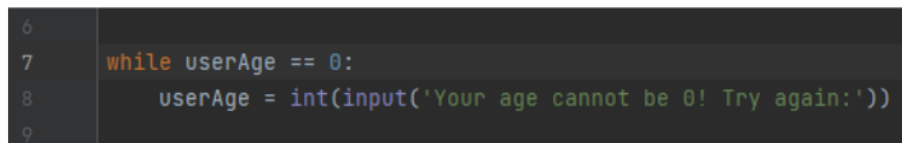
No problems in 8 Exercise 1.py

8 Exercise 1.py
1 userName = str(input('Please state your name: '))
2 userAge = int(input('Please enter your age: '))
3 userHeight = float(input('Please enter your height (in metres): '))
4
5 while userHeight >= 3:
6     userHeight = int(input('Is this correct? That's very high! Input again: '))
7
8 print(userName, ' - ', userAge, ' years old - ', userHeight, ' m tall', sep='')
9
10
11

C:\Users\jonty\OneDrive\Documents\1. Deep Docs\1. Work\Uni\Gaynor\venv\Scripts\python.exe "C:/Users/jonty/OneDrive/Documents/1.
Please state your name: J
Please enter your age: 1
Please enter your height (in metres): 1
Is this correct? That's very high! Input again: 1
Is this correct? That's very high! Input again: 1
J - 1 years old - 1.0m tall

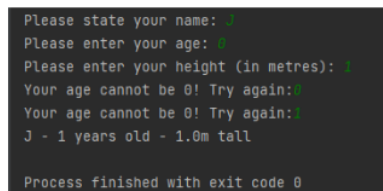
Process finished with exit code 0
```

The author decided to start the loops off with the height loop that prevents the user from inputting a height that is very high, and they still weren't very happy looking at the end result so they added a [sep=""] at the end to get it tidy.



```
6
7 while userAge == 0:
8     userAge = int(input('Your age cannot be 0! Try again:'))
9
```

The age loop was very simple like the height loop, just to simple prevent someone inputting an age that seems unordinary and prompt the user to input again.



```
Please state your name: J
Please enter your age: 1
Please enter your height (in metres): 1
Your age cannot be 0! Try again: 1
Your age cannot be 0! Try again: 1
J - 1 years old - 1.0m tall

Process finished with exit code 0
```

With this the author was happy with how the end result was presented and how the variable inputs were being asked.

Exercise 2:

Allow the user to enter the dimensions of shapes shown below (in cm), then calculate the area of the blue shaded part. Note: The square and rectangular areas are empty.

Note:

Area of circle is given by $A = \pi r^2$

Area of square and rectangle is given by $A = l * w$

```
print('Please enter values in cm')    # specify to the user what units the program uses

circRad = float(input('Please enter the radius of your circle:'))

internalSqu = float(input('Please give the side of the square in your circle:'))
while internalSqu > (circRad * 1.5):
    internalSqu = float(input("That square wouldn't fit inside the Circle! please enter again:"))
print("")
squareArea = internalSqu * internalSqu    # once the square side has been accepted calculates area
and stores

internalReqW = float(input('Please give the width of the rectangle inside the circle:'))
while internalReqW > (circRad * 1.5):
    internalReqW = float(input("That width wouldn't fit inside the Circle! please enter again:"))
print("")

internalReqL = float(input('and please give the length of the rectangle:'))
while internalReqL > (circRad * 1.5):
    internalReqL = float(input("That length wouldn't fit inside the Circle! please enter again:"))
print("")
reqArea = internalReqL * internalReqW    # same as above, once both values validated calc area

circAreaOverall = 3.141 * (circRad * circRad)    # took pi to 3 dp for accuracy

circAreaEnd = circAreaOverall - (squareArea + reqArea)

print('the shaded area of the circle is: ', circAreaEnd, 'cm', sep="")    # sep to make result tidy
```

```
1
2
3     print('the shaded area of the circle is:', )
4
```

The author started with the end line of code as this was probably the easiest

```
1     circRad = float(input('Please enter the radius of your circle:'))
2     internalSqu = float(input('Please give the side of the square in your circle:'))
3     internalReqW = float(input('Please give the width of the rectangle inside the circle:'))
4     internalReqL = float(input('and please give the length of the rectangle:'))
5
6
7
8     print('the shaded area of the circle is:', circAreaEnd)
9
```

They then stated all variables along with the text print for the user, as well as create the ending variable for the result

```

1 circRad = float(input('Please enter the radius of your circle:'))
2 internalSqu = float(input('Please give the side of the square in your circle:'))
3 internalReqW = float(input('Please give the width of the rectangle inside the circle:'))
4 internalReqL = float(input('and please give the length of the rectangle:'))
5
6 circAreaOvall = 3.141 * (circRad * circRad)
7
8 while internalSqu > (circRad * 2):
9     internalSqu = float(input("That wouldn't fit inside the Circle! please enter again:"))
10
11 while internalReqW > (circRad * 2):
12     internalReqW = float(input("That wouldn't fit inside the Circle! please enter again:"))
13
14 while internalReqL > (circRad * 2):
15     internalReqL = float(input("That wouldn't fit inside the Circle! please enter again:"))
16
17 print('the shaded area of the circle is:', circAreaEnd)
18

```

The author then added while loops for the inputs to act as safety nets in case the user inputs numbers that don't work with the circle's radius, as well as adding the equation for the circles overall radius.

```

8 while internalSqu > (circRad * 2):
9     internalSqu = float(input("That wouldn't fit inside the Circle! please enter again:"))
10 squareArea = internalSqu * 2
11
12 while internalReqW > (circRad * 2):
13     internalReqW = float(input("That wouldn't fit inside the Circle! please enter again:"))
14
15 while internalReqL > (circRad * 2):
16     internalReqL = float(input("That wouldn't fit inside the Circle! please enter again:"))
17 reqArea = internalReqL * internalReqW
18

```

They then added the equations for the square and rectangle, however they didn't realise the error in the square equation until later

```

Please enter the radius of your circle:1
Please give the side of the square in your circle:2
Please give the width of the rectangle inside the circle:2
and please give the length of the rectangle:3
That wouldn't fit inside the Circle! please enter again:2
Traceback (most recent call last):
  File "C:\Users\jonty\OneDrive\Documents\1. Deep Docs\1. Wo
    print('the shaded area of the circle is:', circAreaEnd)
NameError: name 'circAreaEnd' is not defined

Process finished with exit code 1

```

The authors testing seemed to be going well

```

1 print('Please enter values in cm')
2
3 circRad = float(input('Please enter the radius of your circle:'))
4 internalSqu = float(input('Please give the side of the square in your circle:'))
5 internalReqW = float(input('Please give the width of the rectangle inside the circle:'))
6 internalReqL = float(input('and please give the length of the rectangle:'))
7
8 circAreaOverall = 3.141 * (circRad * circRad)
9
10 while internalSqu > (circRad * 1.5):
11     internalSqu = float(input('That square wouldn't fit inside the Circle! please enter again:'))
12
13 squareArea = internalSqu * internalSqu
14
15 while internalReqW > (circRad * 1.5):
16     internalReqW = float(input('That width wouldn't fit inside the Circle! please enter again:'))
17
18 while internalReqL > (circRad * 1.5):
19     internalReqL = float(input('That length wouldn't fit inside the Circle! please enter again:'))
20
21 reqArea = internalReqL * internalReqW

```

The author then realised the clarity of the text prints wasn't clear enough (for debugging purposes and for the user's sake, and during this re-read they realised that the squareArea formula was incorrect and changed it.

```

Please enter values in cm
Please enter the radius of your circle:1
Please give the side of the square in your circle:1
Please give the width of the rectangle inside the circle:1
and please give the length of the rectangle:1
That square wouldn't fit inside the Circle! please enter again:1
That width wouldn't fit inside the Circle! please enter again:1
That length wouldn't fit inside the Circle! please enter again:1
the shaded area of the circle is: 1.141cm

Process finished with exit code 0

```

Testing was much clearer now that it specified which value was too large

```

1 print('Please enter values in cm')
2
3 circRad = float(input('Please enter the radius of your circle:'))
4
5 internalSqu = float(input('Please give the side of the square in your circle:'))
6 while internalSqu > (circRad * 1.5):
7     internalSqu = float(input('That square wouldn't fit inside the Circle! please enter again:'))
8     print('')
9 squareArea = internalSqu * internalSqu
10
11 internalReqW = float(input('Please give the width of the rectangle inside the circle:'))
12 while internalReqW > (circRad * 1.5):
13     internalReqW = float(input('That width wouldn't fit inside the Circle! please enter again:'))
14     print('')
15 internalReqL = float(input('and please give the length of the rectangle:'))
16 while internalReqL > (circRad * 1.5):
17     internalReqL = float(input('That length wouldn't fit inside the Circle! please enter again:'))
18     print('')
19
20 circAreaOverall = 3.141 * (circRad * circRad)
21
22 reqArea = internalReqL * internalReqW
23
24 circAreaEnd = circAreaOverall - (squareArea + reqArea)
25
26 print('the shaded area of the circle is: ', circAreaEnd, 'cm', sep='')

```

For the sake of run order and a slight bit of tidiness in the dialogue box, the author adjusted the lines of the while loops to loops straight after the variable is input, as well as enter line breaks.

```
Please enter values in cm
Please enter the radius of your circle:1
Please give the side of the square in your circle:2
That square wouldn't fit inside the Circle! please enter again:1

Please give the width of the rectangle inside the circle:1

and please give the length of the rectangle:1

the shaded area of the circle is: 1.141cm

Process finished with exit code 0
```

They were happy with the end result

Exercise 3:

Write a program that allows the user to enter their car mileage (per day) for a week. After the daily mileage has been entered, calculate the total mileage. Finally, calculate the total cost of petrol for the week, at the rate of £1.35/litre and 11.2 miles to the litre. Use a list and loop(s) where appropriate.

```
miles = []

miles.append(float(input('Please enter your miles for Monday: ')))
while miles[0] < 0:
    miles[0] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Tuesday: ')))
while miles[1] < 0:
    miles[1] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Wednesday: ')))
while miles[2] < 0:
    miles[2] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Thursday: ')))
while miles[3] < 0:
    miles[3] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Friday: ')))
while miles[4] < 0:
    miles[4] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Saturday: ')))
while miles[5] < 0:
    miles[5] = float(input('Sorry, try again:'))

miles.append(float(input('Please enter your miles for Sunday: ')))
while miles[6] < 0:
    miles[6] = float(input('Sorry, try again:'))

print("")

print('Daily miles:', miles)
print("")

weekTotal = float(miles[1] + miles[2] + miles[3] + miles[4] + miles[5] + miles[6] + miles[0])
petrolCost = (weekTotal/11.2)*1.35

print('your total miles for the week are:', weekTotal)
print('The petrol price for the week is:', round(petrolCost, 2))

print('End of program')
```

```
1  miles = []
2
```

Author started by defining the list

```

1  miles = []
2
3  miles.append(float(input('Please enter your miles for Monday: ')))
4  miles.append(float(input('Please enter your miles for Tuesday: ')))
5  miles.append(float(input('Please enter your miles for Wednesday: ')))
6  miles.append(float(input('Please enter your miles for Thursday: ')))
7  miles.append(float(input('Please enter your miles for Friday: ')))
8  print('')
9

```

They then added the list inserts, they did however only add 5 as they thought it was just for the working week.

```

1  miles = []
2
3  miles.append(float(input('Please enter your miles for Monday: ')))
4  miles.append(float(input('Please enter your miles for Tuesday: ')))
5  miles.append(float(input('Please enter your miles for Wednesday: ')))
6  miles.append(float(input('Please enter your miles for Thursday: ')))
7  miles.append(float(input('Please enter your miles for Friday: ')))
8  print('')
9
10 print('Daily miles:', miles)
11 print('')
12
13 weekTotal = float(miles[1] + miles[2] + miles[3] + miles[4] + miles[0])
14 print('your total miles for the week are:', weekTotal)
15 print('End of program')
16

```

Then they coded the end display and the internal total

```

3  miles.append(float(input('Please enter your miles for Monday: ')))
4  while miles[0] < 0:
5      miles[0] = float(input('Sorry, try again:'))
6  |
7  miles.append(float(input('Please enter your miles for Tuesday: ')))
8  miles.append(float(input('Please enter your miles for Wednesday: ')))
9  miles.append(float(input('Please enter your miles for Thursday: ')))
10 miles.append(float(input('Please enter your miles for Friday: ')))
11 print('')
12
13 print('Daily miles:', miles)
14 print('')
15
16 weekTotal = float(miles[1] + miles[2] + miles[3] + miles[4] + miles[0])
17 print('your total miles for the week are:', weekTotal)
18 print('End of program')
19

```

The author then moved onto adding the user safety nets in case of a negative number being input

```

3 miles.append(float(input('Please enter your miles for Monday: ')))
4 while miles[0] < 0:
5     miles[0] = float(input('Sorry, try again:'))
6
7 miles.append(float(input('Please enter your miles for Tuesday: ')))
8 while miles[1] < 0:
9     miles[1] = float(input('Sorry, try again:'))
10
11 miles.append(float(input('Please enter your miles for Wednesday: ')))
12 while miles[2] < 0:
13     miles[2] = float(input('Sorry, try again:'))
14
15 miles.append(float(input('Please enter your miles for Thursday: ')))
16 while miles[3] < 0:
17     miles[3] = float(input('Sorry, try again:'))
18
19 miles.append(float(input('Please enter your miles for Friday: ')))
20 while miles[4] < 0:
21     miles[4] = float(input('Sorry, try again:'))
22 print('')
23
24 print('Daily miles:', miles)
25 print('')
26

```

Once that was added they finished up the rest of the append lines

```

28 petrolCost = (weekTotal/11.2)*1.35

```

They then added the formula for the total cost of the petrol used

```

Please enter your miles for Monday: 3
Please enter your miles for Tuesday: -2
Sorry, try again: 4
Please enter your miles for Wednesday: 50
Please enter your miles for Thursday: 23
Please enter your miles for Friday: 62

Daily miles: [3.0, 4.0, 50.0, 23.0, 62.0]

your total miles for the week are: 142.0
The petrol price for the week is: 17.11607142857143
End of program

Process finished with exit code 0

```

It was this point when they were validating the code against the work example that they noticed the end values were wrong, so they added in the missing days

```
Please enter your miles for Monday: 3
Please enter your miles for Tuesday: -2
Sorry, try again: 4
Please enter your miles for Wednesday: 50
Please enter your miles for Thursday: 23
Please enter your miles for Friday: 62
Please enter your miles for Saturday: 92
Please enter your miles for Sunday: 12

Daily miles: [3.0, 4.0, 50.0, 23.0, 62.0, 92.0, 12.0]

your total miles for the week are: 246.0
The petrol price for the week is: 29.65178571428572
End of program

Process finished with exit code 0
```

With this done and the values correct they were happy with the result, however in the final code they rounded the petrol price to 2 points.

```
40 print('The petrol price for the week is:', round(petrolCost, 2))
41
42 print('End of program')
43
```

Exercise 4:

Pairs

You may implement the rules simply or in a more complex way. For example, you could have the same board each time, or you could set it up so that the values are placed randomly. Use a list to store the content of the game board.

You should use at least one function in your program. Extra credit will be given for more complex implementations.

Note: Remember, there are many marks available for design etc. It is better to have a version of the game that is partially functional, even if it does not incorporate the more involved requirements (eg counting/limiting number of turns or updating the board).

```
import random
print("Welcome to my pairs game! find all 5 pairs!")
print("Here is the board:")

board = ['?', '?', '?', '?', '?', '?', '?', '?', '?', '?']

def printboard():
    print("1 2 3 4 5 6 7 8 9 10")
    for i in range(0, 10):
        print(board[i] + ' ', end="")
    print("")

pairs = ['bottle', 'bottle', 'coin', 'coin', 'key', 'key', 'brush', 'brush', 'hat', 'hat']

# random generator and validate
num0 = random.randint(0, 9)
num1 = random.randint(0, 9)
while num1 == num0:
    num1 = random.randint(0, 9)

num2 = random.randint(0, 9)
while num2 == num0 or num2 == num1:
    num2 = random.randint(0, 9)

num3 = random.randint(0, 9)
while num3 == num0 or num3 == num1 or num3 == num2:
    num3 = random.randint(0, 9)

num4 = random.randint(0, 9)
while num4 == num0 or num4 == num1 or num4 == num2 or num4 == num3:
    num4 = random.randint(0, 9)

num5 = random.randint(0, 9)
while num5 == num0 or num5 == num1 or num5 == num2 or num5 == num3 or num5 == num4:
    num5 = random.randint(0, 9)

num6 = random.randint(0, 9)
while num6 == num0 or num6 == num1 or num6 == num2 or num6 == num3 or num6 == num4 or num6 == num5:
    num6 = random.randint(0, 9)

num7 = random.randint(0, 9)
```

```

while num7 == num0 or num7 == num1 or num7 == num2 or num7 == num3 or num7 == num4 or
num7 == num5 or num7 == num6:
    num7 = random.randint(0, 9)

num8 = random.randint(0, 9)
while num8 == num0 or num8 == num1 or num8 == num2 or num8 == num3 or num8 == num4 or
num8 == num5 or num8 == num6 \
    or num8 == num7:
    num8 = random.randint(0, 9)

num9 = random.randint(0, 9)
while num9 == num0 or num9 == num1 or num9 == num2 or num9 == num3 or num9 == num4 or
num9 == num5 or num9 == num6 \
    or num9 == num7 or num9 == num8:
    num9 = random.randint(0, 9)

print(num0, num1, num2, num3, num4, num5, num6, num7, num8, num9)

hideLabels = []

hidden0 = pairs[num0]
hideLabels.append(hidden0)
hidden1 = pairs[num1]
hideLabels.append(hidden1)
hidden2 = pairs[num2]
hideLabels.append(hidden2)
hidden3 = pairs[num3]
hideLabels.append(hidden3)
hidden4 = pairs[num4]
hideLabels.append(hidden4)
hidden5 = pairs[num5]
hideLabels.append(hidden5)
hidden6 = pairs[num6]
hideLabels.append(hidden6)
hidden7 = pairs[num7]
hideLabels.append(hidden7)
hidden8 = pairs[num8]
hideLabels.append(hidden8)
hidden9 = pairs[num9]
hideLabels.append(hidden9)

def guess(g1, g2):
    if hideLabels[g1-1] == hideLabels[g2-1]:
        board[g1-1] = 'x'
        board[g2-1] = 'x'
        print('hit')
        printboard()
    else:
        print("Incorrect, try again")
        printboard()

printboard()

while '?' in board:
    g1 = int(input('Please enter first guess: '))
    g2 = int(input('Please enter second guess: '))
    guess(g1, g2)

```

```
print('Well done! you have finished the game')
```

```
1 print("Welcome to my pairs game! find all 5 pairs!")
2 print('Here is the board:')
3 print("0 1 2 3 4 5 6 7 8 9")
4
5 |
```

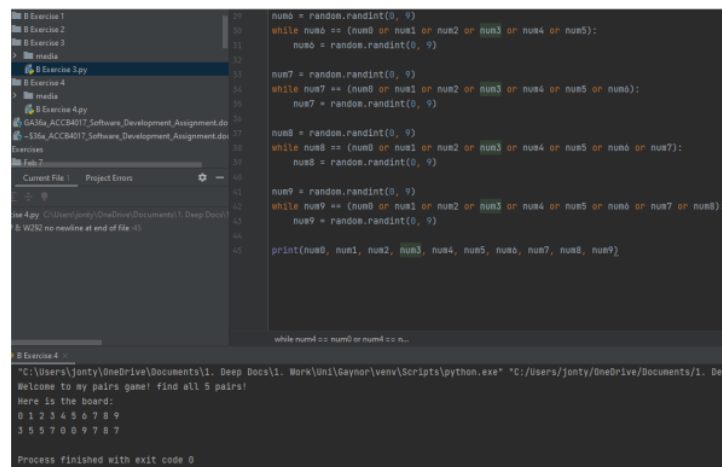
To start with, the author created the introduction for the user by writing print functions

```
1 import random
2 print("Welcome to my pairs game! find all 5 pairs!")
3 print('Here is the board:')
4 print("0 1 2 3 4 5 6 7 8 9")
5
6 pairs = [bottle, bottle, coin, coin, key, key, brush, brush, hat, hat]
7
```

They then wrote out the list of pairs for the program to use and assign randomly, however at the time didn't save them in string format

```
13
14
15 pairs = ['bottle', 'bottle', 'coin', 'coin', 'key', 'key', 'brush', 'brush', 'hat', 'hat']
16
```

This was then corrected by the author



```
20 num0 = random.randint(0, 9)
21 while num0 == (num0 or num1 or num2 or num3 or num4 or num5):
22     num0 = random.randint(0, 9)
23
24 num7 = random.randint(0, 9)
25 while num7 == (num0 or num1 or num2 or num3 or num4 or num5 or num6):
26     num7 = random.randint(0, 9)
27
28 num8 = random.randint(0, 9)
29 while num8 == (num0 or num1 or num2 or num3 or num4 or num5 or num6 or num7):
30     num8 = random.randint(0, 9)
31
32 num9 = random.randint(0, 9)
33 while num9 == (num0 or num1 or num2 or num3 or num4 or num5 or num6 or num7 or num8):
34     num9 = random.randint(0, 9)
35
36 print(num0, num1, num2, num3, num4, num5, num6, num7, num8, num9)
37
38 while num0 == num0 or num4 == num4:
```

Process finished with exit code 0

The author tried to create a block of code that accurately assigned a number from 0-9 without any repeats, however their initial method didn't go to plan

```

29 num0 = random.randint(0, 9)
30 while num0 == num0 or num0 == num1 or num0 == num2 or num0 == num3 or num0 == num4 or num0 == num5:
31     num0 = random.randint(0, 9)
32
33 num7 = random.randint(0, 9)
34 while num7 == num0 or num7 == num1 or num7 == num2 or num7 == num3 or num7 == num4 or num7 == num5 or num7 == num6:
35     num7 = random.randint(0, 9)
36
37 num8 = random.randint(0, 9)
38 while num8 == num0 or num8 == num1 or num8 == num2 or num8 == num3 or num8 == num4 or num8 == num5 or num8 == num6\
39     or num8 == num7:
40     num8 = random.randint(0, 9)
41
42 num9 = random.randint(0, 9)
43 while num9 == num0 or num9 == num1 or num9 == num2 or num9 == num3 or num9 == num4 or num9 == num5 or num9 == num6\
44     or num9 == num7 or num9 == num8:
45     num9 = random.randint(0, 9)
46
47 print(num0, num1, num2, num3, num4, num5, num6, num7, num8, num9)

```

They decided the best way to do it would unfortunately be the longest, which assigned the numbers without repeats.

```

46
47 print(num0, num1, num2, num3, num4, num5, num6, num7, num8, num9)
48
49 hidden0 = pairs[num0]
50 hidden1 = pairs[num1]
51 hidden2 = pairs[num2]
52 hidden3 = pairs[num3]
53 hidden4 = pairs[num4]
54 hidden5 = pairs[num5]
55 hidden6 = pairs[num6]
56 hidden7 = pairs[num7]
57 hidden8 = pairs[num8]
58 hidden9 = pairs[num9]
59 |

```

They then started to use the random numbers (numX) with the list of (pairs[]) to 10 new variables.

```

60
61 hideLabels = []
62
63 hideLabels.append(hidden0 == pairs[num0])
64 hidden1 = pairs[num1]
65 hidden2 = pairs[num2]
66 hidden3 = pairs[num3]
67 hidden4 = pairs[num4]
68 hidden5 = pairs[num5]
69 hidden6 = pairs[num6]
70 hidden7 = pairs[num7]
71 hidden8 = pairs[num8]
72 hidden9 = pairs[num9]
73 print('? ? ? ? ? ? ? ? ?')
74
75 guess1 = int(input('Please enter first guess: '))
76 guess2 = int(input('Please enter second guess: '))
77 guesses(guess1, )
78

```

The author then tried to add these new variables into a list, essentially the same as the original pairs[] list, but reassigned in a random order, Python however didn't allow this. They also began to write code on line 77 for a later function.

```

59 hideLabels = []
60
61 hidden0 = pairs[num0]
62 hideLabels.append(hidden0)
63 hidden1 = pairs[num1]
64 hideLabels.append(hidden1)
65 hidden2 = pairs[num2]
66 hideLabels.append(hidden2)
67 hidden3 = pairs[num3]
68 hideLabels.append(hidden3)
69 hidden4 = pairs[num4]
70 hideLabels.append(hidden4)
71 hidden5 = pairs[num5]
72 hideLabels.append(hidden5)
73 hidden6 = pairs[num6]
74 hideLabels.append(hidden6)
75 hidden7 = pairs[num7]
76 hideLabels.append(hidden7)
77 hidden8 = pairs[num8]
78 hideLabels.append(hidden8)
79 hidden9 = pairs[num9]
80 hideLabels.append(hidden9)

```

To circumvent this the author had to add a new line of code under each variable assigning it into the hideLabels[] list.

```

2
3
4 def guesses(est1, hide1):
5     ans1 = 0
6     ans2 = 0
7     if est1 == hide1:
8         ans1 = 1
9     else:
10        ans1 = 0
11    return ans1
12

```

This function essentially was an attempt at giving a variable 2 values, the random number (hide1 and hide2) and a state of being a match or not (ans1 and ans2 = 0 or 1)

```

4
5 board = ['?', '?', '?', '?', '?', '?', '?', '?', '?', '?']
6
7
8 def printboard():
9     print("1 2 3 4 5 6 7 8 9 10")
10    for i in range(0, 10):
11        print(board[i] + ' ', end='')
12    print('')
13

```

After that idea didn't go anywhere, the author decided to rewrite the board print code as a list, and change the contents as such.

```

83 def guess(g1, g2):
84     if hideLabels[g1-1] == hideLabels[g2-1]:
85         board[g1-1] = 'x'
86         board[g2-1] = 'x'
87         print('hit')
88         printboard()
89     else:
90         print("Incorrect, try again")
91         printboard()
92
93
94     printboard()
95
96 while '?' in board:
97     g1 = int(input('Please enter first guess: '))
98     g2 = int(input('Please enter second guess: '))
99     guess(g1, g2)
100    print('Well done! you have finished the game')
101

```

After this the author finished off the code with a working way to compare variables and tell the user that they got a pair.

Software Dev Assignment 1

GRADEMARK REPORT

FINAL GRADE

61 / 100

GENERAL COMMENTS

Instructor

PART A

Variables - You have talked about variables and alluded to types, but you need to say what the types are ie int, float string. You have talked about the way that Python infers the variable type but you need more on how/why these types are considered dynamic (in Python, types can change within the lifetime of the program, for example, when dividing two integers, the resulting type is changed to float) 3/5

If/Else -You've got the main points, could also have talked about and/or multiple conditions. 4/5

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

PART B

Exercise 1

Logical Stages-Testing - No logical stages shown? Please see accompanying "How to lay out your assignment Part B" document. I think you've misunderstood the testing stage - again, see document. 1.5/5

Coding-Annotation - A working program, but you need validation here to ensure that the user does not enter out of range age/height etc. (ie reasonable range of

human life span/height) You've only tested that age isn't 0 (it could be a minus number according to your code) and only the upper limit of height 6/10

Exercise 2

Logical Stages-Testing - Same comments as above, testing is better here 2/5

Coding-Annotation - Generally good program, you consider size of circle against the square/rectangle, could also consider size of square/rectangle against each other. Also validate for negative numbers 7.5/10

Exercise 3

Logical Stages-Testing - Same comments as above 6/10

Coding-Annotation - You've got a list, but then you haven't used a loop (therefore negating the reasons for having a list!) You could have held the days of the week in a list too, which would have meant you could loop. You could also have validated for the upper milages and used sum() to add the values 5.5/10

Exercise 4

Logical Stages-Testing - No real logical stages, testing needs to show final outputs. 1 /10

Coding-Annotation - You could use random.shuffle() to simplify your random routine. Some good use of functions - be careful to validate the users guess to avoid list overflow and to make sure they don't select the same number for both inputs (because it would match on itself) A good effort overall 19/25

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.

You need to avoid using active tenses in your academic written work such as "they" and "The author". You need to use the passive voice throughout. For example "The author decided to start the loops off" -> write instead "The loops were started with...

Assignment overall: 60.5%

Test mark: 72%

OVERALL MARK FOR MODULE: 64%

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
