Computer Fundamentals Assignment Questions 1-4

PORTFOLIO 2

Phil O’Donnell

2020-2021

# **Introduction**

Prior to this Bootcamp, I have self-taught some fundamentals of Java. With the exception of Functions and writing to files I was quite familiar with the basic concepts and algorithmic thinking required to write code which hopefully adapted well to Python. I’m very happy that I learned Java before python, as I found python syntax to be more enjoyable to learn than Java.

I have completed each task the best of my ability and have tried to exceed the taught concepts where possible.

I have also added notes and descriptions to most lines of code to hopefully show that I have fully grasped the code before implementing it. I hope you enjoy reading my answers.

**Portfolio Question 1: Calculate Age from Year of Birth**

Here we have to ask the user for their age which we convert to type ‘int’ and assign to a variable, which in this code was ‘YoB’. We then deduct the value from the current year which we then print to show the users’ age. I wanted to demonstrate that I knew how to insert the user’s age in between 2 strings when printing “You are”, age, “years old!”

Upon reflection I could have added data validation to ensure that the user input a valid year.

**Code:**

'''

Request user's year of birth and convert

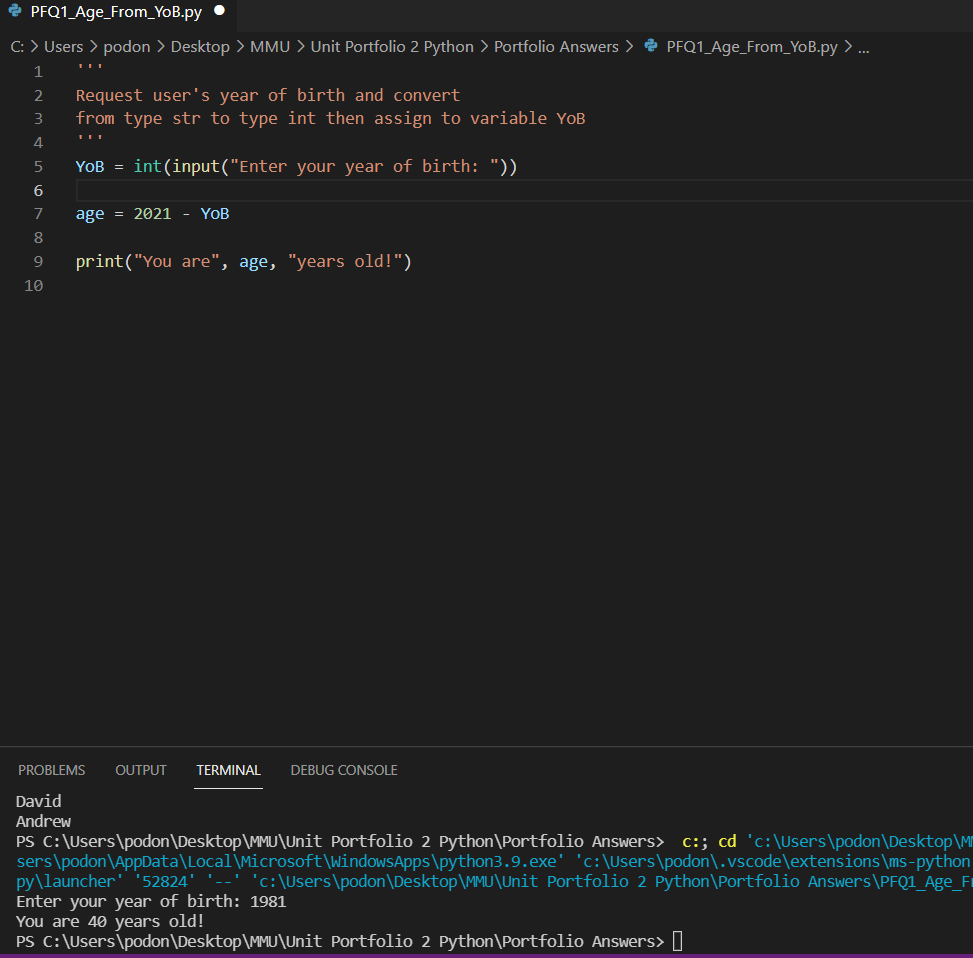
from type str to type int then assign to variable YoB

'''

YoB = int(input("Enter your year of birth: "))

age = 2021 - YoB

print("You are", age, "years old!")



**Portfolio Question 2 : Student gets 10% discount**

Here we wrote code which asks the user if they are a student or staff member and assigns then value to the variable ‘status’. I learnt how to avoid case sensitivity and implemented it to the code which then applies a 10% discount if the user is a student and prints how much they have to pay.

Unfortunately, staff members don’t get a discount, but the program does recognise them as staff, then reminds them they get no discount!

The final decision is for all other inputs who aren’t student or staff and also receive no discount.

I could have nested the final decision which I have commented in the code, but thought this to be unnecessary.

**Code:**

# request the item cost and convert to a float

cost = float(input("Enter the item cost: £"))

#request user status and convert to lower case incase the user's input contains different cases

status = input("Enter your status ('student' or 'staff'):").lower()

if(status == "student"):

discount = 0.1

print("Amount to pay including 10% student discount: £", cost - cost \* discount)

elif(status == "staff"):

print("Amount to pay: £", cost, "No staff discount") #inputting the cost between 2 strings optional

else:

print("Unknown user - no discount. Please pay: £", cost)

'''

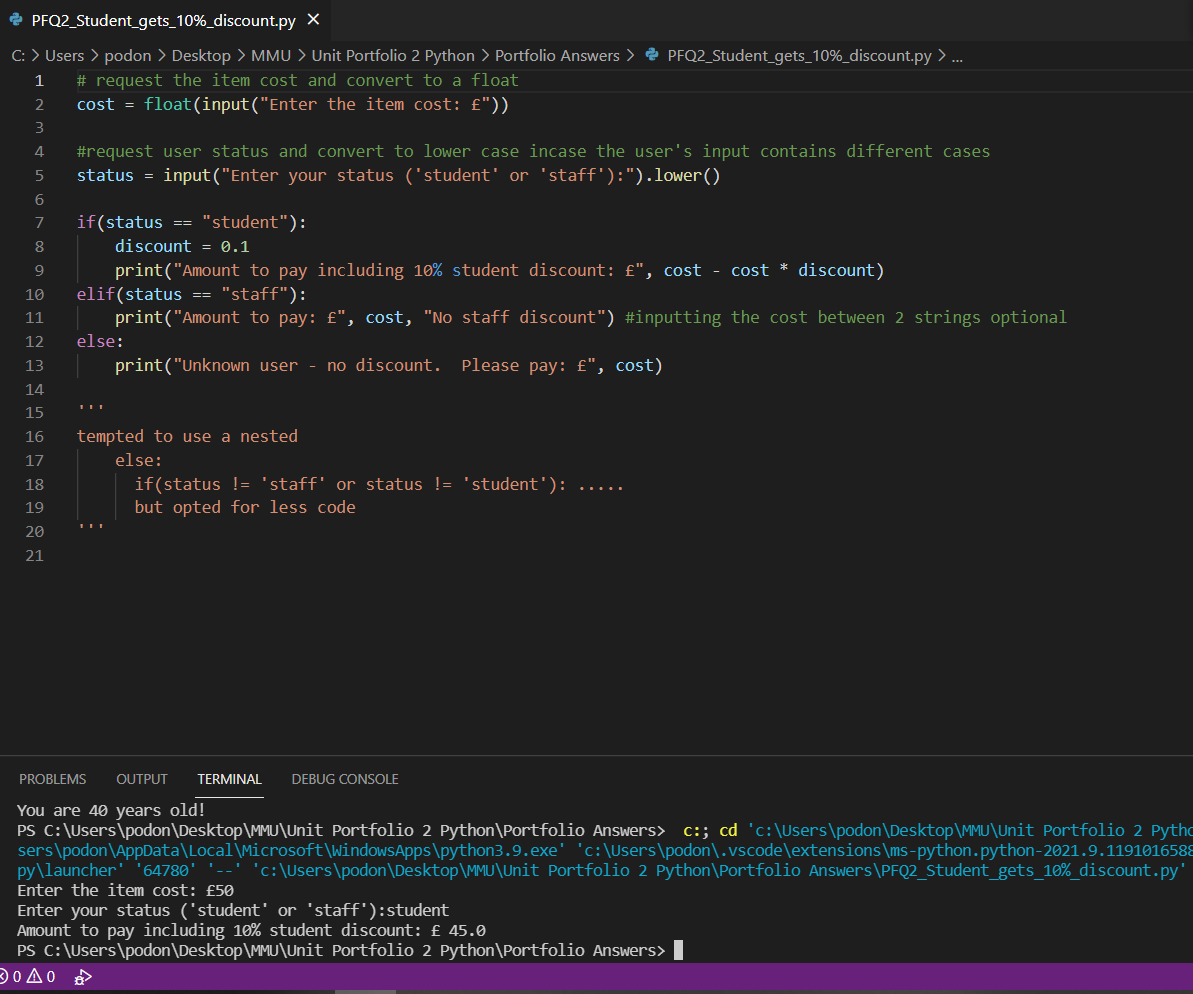
tempted to use a nested

else:

if(status != 'staff' or status != 'student'): .....

but opted for less code

'''



**Portfolio Question 3: Simple Login System**

In this program we had to ask the user for a username and password which ran on a continuous while loop until both the username and password meet the desired criteria where username = ‘admin’ and password = ‘pwd’. I chose to opt to convert the user’s input to lowercase in case they input the right word, but wrong case.

I chose a while loop because we don’t know how many times the user will input the wrong login data. I could improve this algorithm by adding a count variable that increments by 1 after each unsuccessful attempt within the while loop that will only iterates whilst count is less than or equal to 3:

**Code:**

username = input("Enter your User Name:").lower() # convert each input to lowercase so it's not case sensitive

password = input("Enter your password:").lower()

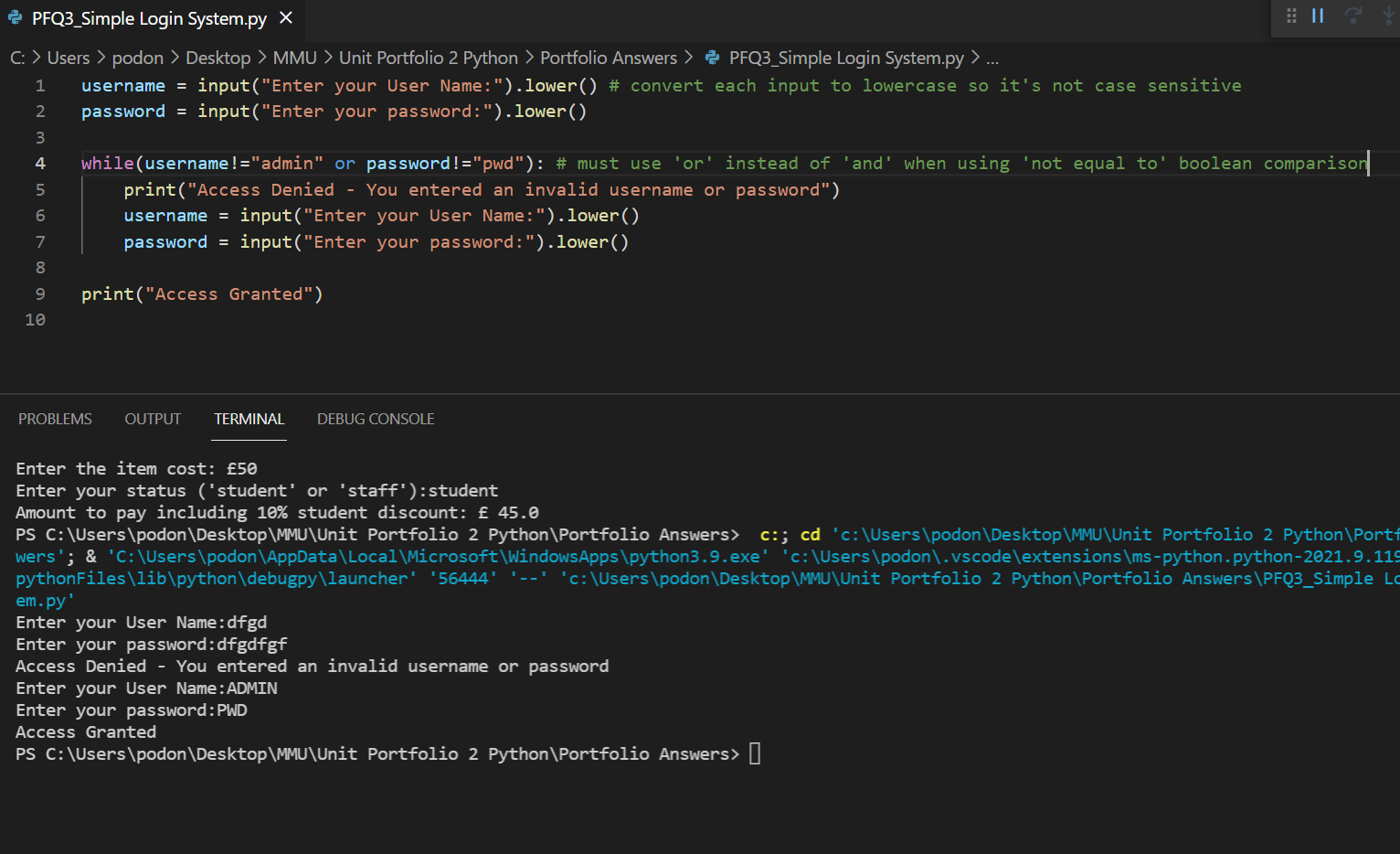
while(username!="admin" or password!="pwd"): # must use 'or' instead of 'and' when using 'not equal to' boolean comparison

print("Access Denied - You entered an invalid username or password")

username = input("Enter your User Name:").lower()

password = input("Enter your password:").lower()

print("Access Granted")



**Portfolio Question 4: Honours Classification**

**In this portfolio question we were introduced to functions which I have covered in Javascript, but always found a bit confusing, in particular the return statement, however I was determined to grasp them and I have hopefully demonstrated a better understanding. I found there to be two different ways to work with functions and couldn’t decide which was best, so have includes both sets of code below. I still need to work on functions to eventually master them, but this is definitely a step in the right direction:**

**Code:**

'''

I've hopefully answered this correctly 2 ways. The first algorithm demonstrates the return

function and below in comments I have written the same program but without the return function,

by passing the argument from the function call to the function parameter

'''

def get\_class():

firstName = input("Enter your first name:")

lastName = input("Enter your last name:")

grade = float(input("Enter your grade:")) ## set to float in case their grade has a decimal

while(grade < 0 or grade > 100): # data validation loop in case they input an impossible grade

print("Error - the grade must be between 0 and 100")

grade = float(input("re-enter your grade:"))

if(grade>= 70): # no need to set the upper limit as the data validation code has that covered

honourC = "First Class"

msg = "Excellent, well done!"

elif(grade >= 60):

honourC = "Upper Second"

msg = "Very good, well done!"

elif(grade >= 50):

honourC = "Lower Second"

msg = "Good, well done!"

elif(grade >= 40):

honourC = "Third Class"

msg = "Could have done better!"

elif(grade >= 35):

honourC = "Pass Degree"

msg = "Work harder next time!"

else:

honourC = "Fail"

msg = "Oh dear, try again!"

print(firstName, "", lastName, "-", honourC)

return grade # stores the grade so it can be assigned to a variable if called upon

def get\_congrats():

# no need for data validation here

if(official\_grade >= 70): # no need for upper limit due to passing validation in previous function

msg = "Excellent, well done!"

elif(official\_grade >= 60):

msg = "Very good, well done!"

elif(official\_grade >= 50):

msg = "Good, well done!"

elif(official\_grade >= 40):

msg = "Could have done better!"

elif(official\_grade >= 35):

msg = "Work harder next time!"

else:

msg = "Oh dear, try again!"

print(msg)

official\_grade = get\_class()

official\_message = get\_congrats()

"""

I would have liked to have both messages occupy the

same line, but if that's possible, I don't know how to do it yet

"""

"""

Alternative method passing the argument to the function parameter

def display\_class(grade):

if(grade>= 70): # no need to set the upper limit as the data validation code has that covered

honourC = "First Class"

msg = "Excellent, well done!"

elif(grade >= 60):

honourC = "Upper Second"

msg = "Very good, well done!"

elif(grade >= 50):

honourC = "Lower Second"

msg = "Good, well done!"

elif(grade >= 40):

honourC = "Third Class"

msg = "Could have done better!"

elif(grade >= 35):

honourC = "Pass Degree"

msg = "Work harder next time!"

else:

honourC = "Fail"

msg = "Oh dear, try again!"

print(firstName, "", lastName, "-", honourC)

def display\_congrats(grade):

if(grade >= 70): # no need for upper limit due to passing validation in main code

msg = "Excellent, well done!"

elif(grade >= 60):

msg = "Very good, well done!"

elif(grade >= 50):

msg = "Good, well done!"

elif(grade >= 40):

msg = "Could have done better!"

elif(grade >= 35):

msg = "Work harder next time!"

else:

msg = "Oh dear, try again!"

print(msg)

firstName = input("Enter your first name:")

lastName = input("Enter your last name:")

grade = float(input("Enter your grade:"))

while(grade < 0 or grade > 100): # data validation loop in case they input an impossible grade

print("Error - the grade must be between 0 and 100")

grade = float(input("re-enter your grade:"))

display\_class(grade)

display\_congrats(grade)

"""

