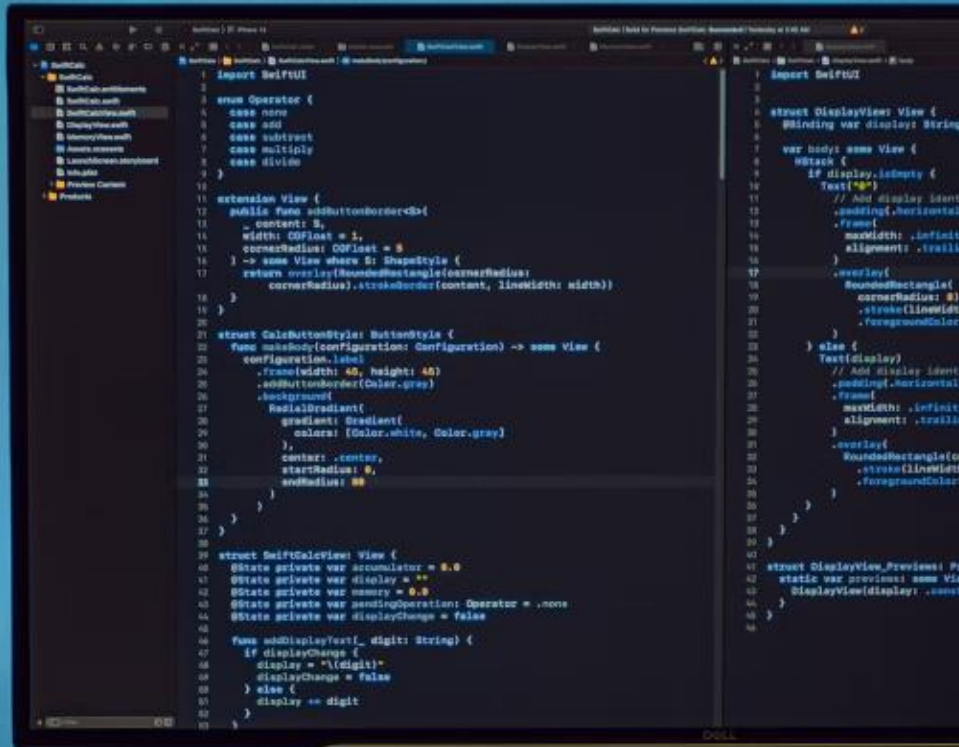


FOOTBALL SKILLS

Niwhar Amin / Software Engineering / Start Date: 02/11/2023



```
import SwiftUI

enum Operator {
    case none
    case add
    case subtract
    case multiply
    case divide
}

extension View {
    public func addButtonBorder<B> (
        _ content: B,
        width: CGFloat = 1,
        cornerRadius: CGFloat = 5
    ) -> some View where B: ShapeStyle {
        return overlay(RoundedRectangle(cornerRadius:
            cornerRadius).strokeBorder(content, lineWidth: width))
    }
}

struct CalcButtonStyle: ButtonStyle {
    func makeBody(configuration: Configuration) -> some View {
        configuration.label
        .frame(width: 40, height: 40)
        .addButtonBorder(Color.gray)
        .background(
            RadialGradient(
                gradient: Gradient(
                    colors: [Color.white, Color.gray]
                ),
                center: .center,
                startRadius: 0,
                endRadius: 40
            )
        )
    }
}

struct SelfCalcView: View {
    @State private var accumulator = 0.0
    @State private var display = ""
    @State private var memory = 0.0
    @State private var pendingOperation: Operator = .none
    @State private var displayChange = false

    func addDisplayFor(_ digit: String) {
        if displayChange {
            display = "\(digit)"
            displayChange = false
        } else {
            display += digit
        }
    }
}
```

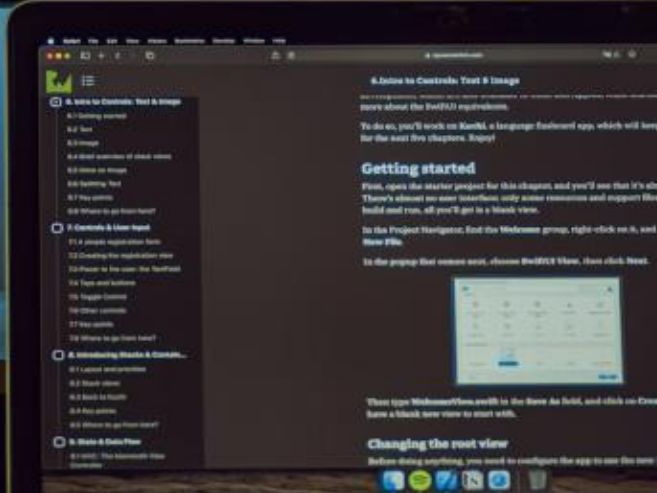


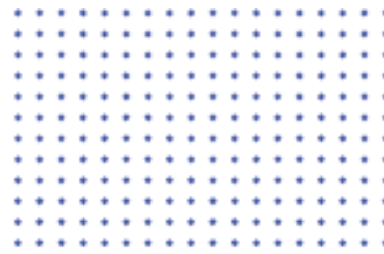


TABLE OF CONTENTS

#AMI23589889 / LAST UPDATE: 29/11/2023

Overview.....	3
Discussion.....	4
Execution Of Code.....	5
Method.....	8
Gantt Chart Reflection.....	10
Evaluation.....	11
References	12

OVERVIEW



This coursework is like an upgraded version of Coursework 1, aiming to get better. In this round, I added some new things like using fancier functions and bringing in outside tools like 'tabulate' and 'datetime.' To figure it all out, I checked out "Computer Programming for Beginners" and watched helpful tutorials on YouTube. The hour-long videos on Moodle were especially helpful, along with tips from ChatGPT. They really helped me understand how tables work and tackle the challenges in this coursework.

The key focus was refining how functions work for each variable and giving them different names. Learning to use arrays for handling salary values was a big step in understanding how data fits together in this context. But as I looked deeper into it, I realized this coursework doesn't quite match up with real business situations. Especially in deciding player salaries, it's just too simplified. It doesn't cover all the things we need to think about in the real business world. These limitations show how different this coursework is from real business situations. It misses out on considering lots of important stuff needed to decide salaries in actual business settings. So, exploring these shortcomings shows the need for a more realistic approach.

We should set up problems and scenarios that better match how things work in real business situations. I want to dig into why this coursework doesn't capture the real complexities of business situations and the many important factors needed for deciding actual salaries. Ultimately, this coursework highlights how important it is to make academic stuff more like what happens in the real world. It helps make what we learn more useful and relevant.

DISCUSSION

The Idea

I wanted to share my thoughts on the coursework by making a PowerPoint presentation to explain my views. After working with the functions and seeing the results, I realized that the inputs and outputs were way off the mark. This being the second coursework, I understand, but it made me think beyond the basics we were given about players.

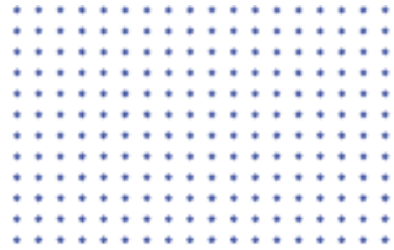
One big issue is that we don't have a specific currency to gauge if the salaries we assigned match the players' real circumstances. Also, we didn't consider the player's influence, like how someone like Cristiano Ronaldo's huge influence could affect their salary due to the attention they attract. I tried to think about these things while working on my coursework and found it unfair to just assign them arbitrary values like "1000 700."

If I had more time, I would've tried adding a feature to fetch actual football player salary data from specific websites. That would've made my code more realistic and reliable. Looking back, this coursework made me realize the need for a more comprehensive approach. We should think about different factors like currency and player influence to better reflect reality when determining player salaries in our coursework.



EXECUTION OF CODE

Screenshots



Importing packages

```
import datetime
from tabulate import tabulate
```

- These allow me to import packages such as datetime, used for the age and D.o.B. Whereas tabulate is used for the table.

Function Lists

```
player_data = []
overall_ratings = {}
salary_ranges = {}
player_ages = {}
```

- This stores the empty lists of each variable which will be used in the functions later.

Range Function

```
def range(inside):
    while True:
        skill = input(inside)
        if skill.isdigit() and 0 <= float(skill) <= 5:
            return float(skill)
        else:
            print("The rating you entered was invalid.")
```

- Function for range of skills that only allows values from 0 – 5. Prints an error message if incorrect.

Player_ID

```
def num_player_id():
    while True:
        player_id = input("\nEnter player's ID (2-digit num) or 'end' to finish: ")
        if player_id.lower() == "end":
            return "end"
        elif len(player_id) != 2 or not player_id.isdigit():
            print("The ID you entered was invalid.")
        else:
            return player_id
```

- Function for player ID and only allows a 2 digit number to be inputted.

D.o.B Function

```
def check_date_of_birth():
    while True:
        dob = input("Enter player's date of birth (YYYY-MM-DD): ")
        try:
            datetime.datetime.strptime(dob, '%Y-%m-%d')
            return dob
        except ValueError:
            print("The date you entered was invalid. Please use the format YYYY-MM-DD.")
```

- Function for D.o.B formatted in ISO which prints an error message if it is not in the correct format.

Calculate Rating

```
def calculate_rating():
    a = range("Enter Their Speed Stats (0-5): ")
    b = range("Enter Their Shooting Stats (0-5): ")
    c = range("Enter Their Passing Stats (0-5): ")
    d = range("Enter Their Defending Stats (0-5): ")
    e = range("Enter Their Dribbling Stats (0-5): ")
    f = range("Enter Their Physicality Stats (0-5): ")

    overall_rating = (a + b + c + d + e + f) * 100 / 30
    return overall_rating
```

- Function for rating that calculates the sum of each skill (*100/30)

Salary Array

```
def calculate_salary(overall_rating):
    salary_values = [1000, 700, 500, 400]

    if overall_rating >= 80:
        return [salary_values[0]]
    elif 60 < overall_rating < 80:
        return [salary_values[0], salary_values[1]]
    elif overall_rating == 60:
        return [salary_values[1]]
    elif 45 < overall_rating <= 60:
        return [salary_values[1], salary_values[2]]
    elif overall_rating == 45:
        return [salary_values[2]]
    elif 30 < overall_rating <= 45:
        return [salary_values[2], salary_values[3]]
    else:
        return [salary_values[3]]
```

- Salary placed in an array to calculate from the given overall rating. It prints depending on the values given within.

Age

```
def calculate_age(date_of_birth):
    dob = datetime.datetime.strptime(date_of_birth, '%Y-%m-%d').date()
    today = datetime.date.today()
    age = today.year - dob.year - ((today.month, today.day) < (dob.month, dob.day))
    return age
```

- Function for age with given package (datetime) that calculates the age by subtracting the D.o.B inputted.

Table

```
def display_table():
    sum_table = []
    for player_id, player_name, dob, overall_rating in sorted(player_data, key=lambda x: x[0]):
        age = str(player_ages.get(player_id, '')).strip()

        salary_range = salary_ranges.get(player_id, '')
        sum_table.append([player_id, player_name, dob, age, overall_rating, salary_range])

    headers = ["UID", "Name", "D.o.B", "Age", "Score", "Salary Range"]
    return tabulate(sum_table, headers=headers)
```

- Displays table with the sum of all the variables that will be outputted into a table.

Main Function

```
def main():
    count = 0
    while count < 3:
        player_id = num_player_id()
        if player_id.lower() == "end":
            break

        player_name = input("Enter player's name: ")
        date_of_birth = check_date_of_birth()
        age = calculate_age(date_of_birth)
        overall_rating = calculate_rating()
        salary_indices = calculate_salary(overall_rating)

        player_data.append((player_id, player_name, date_of_birth, overall_rating))

        overall_ratings[player_id] = overall_rating
        salary_ranges[player_id] = ' '.join(map(str, salary_indices))
        player_ages[player_id] = age

    print("This Player's ID Is: ", player_id)
    print("This Player's Name Is: ", player_name)
    print("This Player's D.o.B Is: ", date_of_birth)
    print("This Player's Age Is: ", age)
    print("This Player's Score Is: ", overall_rating)
    print("This Player's Overall Rating Is: ", round(overall_rating))
    print(f"This Player's Salary Range Is: {' '.join(map(str, salary_indices))}")

    summary = display_table()
    print(summary)

    count += 1
```

- Main function with all the code contained inside. Contains a counter so once 3 players are inputted the code will end.

Saving File

```
def save_files(sum_table):
    file_path = "players.txt"
    with open(file_path, "w") as file:
        file.write(sum_table)
    print(f"Summary saved to {file_path}")
```

- Saves the file as “players.txt” when run.

Calling Main

```
if __name__ == "__main__":
    main()
    summary = display_table()
    print(summary)
    save_files(summary) #calls
```

- Calls main function and summary of table/file.

METHOD

How I executed this

In meandering through the convoluted pathways of this code, I made sure to leave no stone unturned by embedding copious annotations. I delved into the resources available on Moodle, earnestly seeking hidden insights to unravel the complexities. With meticulous attention, I compared this current rendition to the initial coursework, adorning the code with detailed annotations to denote the increased intricacies and alterations made, ensuring clarity for anyone perusing the script. When grappling with table imports and datetime functionalities, my initial refuge was YouTube tutorials, and I diligently chronicled my learning journey within the code's comments.

However, the rigid formatting demanded by the code grader posed a formidable challenge. I tactfully inserted comments to describe how I turned to ChatGPT for guidance, elucidating the way it aided in aligning the code with the stringent formatting prerequisites. The careful articulation in these annotations reflects the considerable time and dedication invested in understanding and implementing the required adjustments to adhere to the code grader's standards.

Reflecting on my endeavors, I acknowledge that managing time more efficiently could have allowed me to incorporate an additional feature enhancing the realism of salary computations. Nevertheless, I take pride in the extensive comments woven throughout the code. Each method, function, and intricate segment is meticulously expounded upon in the annotations, ensuring comprehensibility for any discerning reader. The encapsulation predicament, arising from amalgamating previous coursework into a singular entity, has been addressed through detailed comments outlining plans for future refinement. Specific mentions within these annotations underline the intent to fragment code segments into discrete capsules for enhanced encapsulation and ease of maintenance.

The comments adorning the code exemplify not only my grasp of programming concepts but also the painstaking efforts taken to render the code intelligible and adaptable. The invaluable feedback received from peers and lecturers has profoundly contributed to refining and strengthening both the code and its accompanying report. Their insights have been methodically woven into the comments, enriching the understanding and coherence of the codebase.





METHOD

What I Could Have Added

Due to constraints on my time, I sadly couldn't expand the capabilities of my code as I had hoped. I had considered the idea of integrating a function to procure data from a specialized website dedicated to player salaries. This enhancement would have allowed for a more precise allocation of salaries to individual players, considering their Instagram followers, thus drawing a correlation between their influence and earnings.

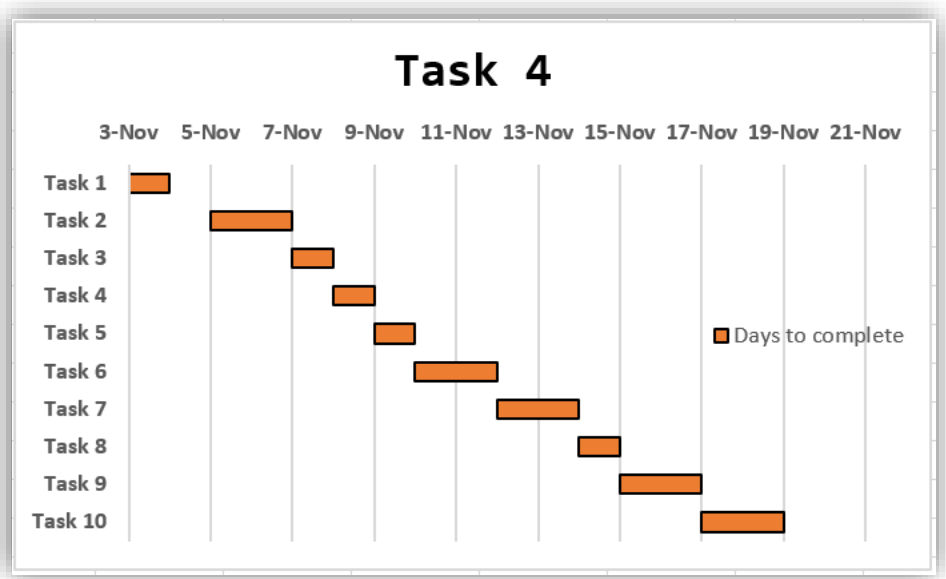
Additionally, the inclusion of currency conversion or displaying various currencies based on geographical locations aimed to highlight the diverse global presence of players. However, achieving these enhancements would have demanded a considerable amount of time, a resource I found to be insufficient. Another thought involved amending the table after its printing by enabling user input to fill gaps and pre-set headers. Despite these potential improvements, I find contentment in the current state of my coursework and the earnest effort I devoted to it.

Football players' earnings reflect more than just the outside view; they embody global symbols, blending talent and cultural resonance. Their influence extends far beyond the pitch, inspiring millions and driving immense revenues to clubs and brands. These athletes, through dedication and sacrifices, channel years of training into performances that captivate worldwide audiences. Their marketability, talents, and role-model status justify substantial earnings, aligning with their profound impact on diverse communities. As icons who entertain, inspire, and shape culture, their remuneration stands as recognition not just of their skills but also of the immense value they bring to the sport and society at large.

GANTT CHART REFLECTION

Regrettably, the Gantt chart I meticulously crafted for coursework 1 did not see fruition as intended. Unforeseen circumstances outside the academic realm caused me to fall behind, compelling an ardent pursuit to regain ground with other module assignments. Despite its non-adherence, the Gantt chart stands as a testament to my conscientious planning, delineating a structured breakdown of tasks into manageable segments.

Each point marked a designated day set aside for completing specific aspects of the coding endeavor. The challenges encountered amid external disruptions impeded the adherence to this well-thought-out schedule. However, the Gantt chart served as a guiding beacon, providing a roadmap for the meticulous segmentation of my coding journey. Though it could not be strictly followed, it remains a poignant reminder of the importance of structured planning and the unforeseen hurdles that can impact our academic pursuits.



(Fig. 1) Gantt chart from my coursework 1 that displays the tasks I assigned to myself to get done for this coursework, broken up into dates that was meant to be followed through.

EVALUATION

The PowerPoint presentation was an attempt to delve deeper into the complexities of determining football player salaries. Initial efforts were thwarted by time constraints, limiting the incorporation of critical elements such as player influence and currency variations. The code, though comprehensive, lacked real-world relevance due to its oversimplified approach in assigning arbitrary values. The coursework, an advancement from a previous iteration, aimed for improvement. Learning from online tutorials, 'Computer Programming for Beginners,' YouTube videos, Moodle resources, and assistance from ChatGPT proved instrumental in navigating challenges.

However, it surfaced the discrepancy between the academic exercise and the multifaceted realities of business scenarios. Reflecting on the process, the realization emerged that the coursework simplifies the complexities inherent in actual business settings. It fails to encompass crucial factors pivotal in determining authentic player salaries, such as market influence and currency dynamics.

Consequently, the coursework highlights the need for a more realistic and holistic approach to academic exercises, bridging the gap between theory and practical application. In an ideal scenario with more time, the inclusion of a function to extract real player salary data from dedicated websites was proposed. This addition would have rendered the code more credible and aligned with genuine market conditions. It underscores the need for coursework to emulate real-world intricacies for a comprehensive learning experience. The evaluation brings to the forefront the coursework's limitations in mirroring business complexities. It highlights the necessity of refining academic exercises to mimic real-world challenges and factors essential in decision-making. Furthermore, it stresses the importance of incorporating variables like player influence and currency dynamics in determining salaries, critical in the global football landscape.

In conclusion, this evaluative reflection underscores the significance of bridging the divide between academic exercises and practical business realities. It advocates for a more comprehensive approach in coursework, emphasizing the incorporation of multifaceted factors crucial in decision-making processes. It emphasizes the need to infuse academic learning with real-world complexities to foster a deeper understanding and relevance in preparing individuals for professional environments.



REFERENCES

[1] <https://www.manning.com/books/hello-world-third-edition>

[2] https://www.w3schools.com/python/python_datetime.asp

[3] https://www.w3schools.com/python/python_file_open.asp

[4] https://www.w3schools.com/python/python_file_write.asp

[5] https://www.w3schools.com/python/python_lambda.asp