Assignment Details

Item	Description[1]
Module Title:	Programming for Analytics
Module Code:	B9BA100
Module Lecturer (s):	Suman Gurbani ,Muhammad Alli, Eugene O'Regan, Muhammad Asad
Programme/Cohort:	MSc in Business Analytics January Intake
Method of Assessment:	Assignment
Percentage (%) Weighting:	50%
MIMLOs being assessed:	2, 4, 5
Assessment Number:	1
Individual/Group:	Individual
Issue Date:	11/02/2025
Submission Date:	16/03/2025
Feedback Date:	
Feedback Strategy:	Moodle

Assignment Task

You are **NOT** allowed to use AI for this assignment.
All questions must be attempted.

The assignment consists of 5 questions. All questions must be attempted. You are required to submit your answers as a single PDF document. Python code must be submitted into the PDF document. It must be clearly marked which code is the answer to which question or question part. Outputs from the code should be made into comments at the end of the code.

Total out of 100 Marks.

1. Write a program to develop real-world tax calculation system that generates Statement of Liability for an employee based on their annual gross income. To generate the SOL, the system must compute the following:
   - Universal Social Charge(USC): It is based on income slabs, as per Table 1.1.
   - Pay Related Social Insurance(PRSI): It is 4.1% of the gross income.
   - Income Tax: It is based on tax brackets, mentioned in Table 1.2.
   - Employees can claim their applicable tax credits from Table 1.3.
   - Final Tax Liability = (USC + PRSI + Income Tax) – Total Tax Credits
   - Net Income = Gross Income – Final Tax Liability.

Tax Calculation Details

Table 1.1: USC Slabs

Threshold	**Tax Rate**
First €12,012	0.5%
Next €15,370	2%
Next €42,662	3%
Remaining Balance after €42,662	8%

Table 1.2: Income Tax Slabs

Annual Income	**Tax Rate**
Up to €44,000	20%
Above €44,000(balance amount)	40%

Table 1.3: Tax Credits

Credit Type	**Amount**
Single Person	€2000
Married/Civil Partner	€4000
Employee	€2000
Dependent(Maximum 2)	€305 per dependent
Rent	€1000

User has to Input the following data:

Input	Constraint	Sample
Employee Id	Mandatory, Unique Identifier	E10180
Employee Name	Alphabets and Spaces	Martin Cox
Gross Income(€)	Must be a positive number	50000
Single	Yes/No	Yes
Married/Civil Partner	Yes/No	No
Employee Credit	Yes/No	Yes
Rent Credit	Yes/No	Yes
No. of Dependents	Must be 0,1,2	1

Once the data has been entered, it should display the Output in this format:

Statement of Liability for Martin Cox

Gross Income: €50,000.00
USC Deduction: €1,046.00
PRSI Deduction: €2,050.00
Income Tax: €11,200.00

Total Tax Credits Applied: €5,305
Final Tax Liability: €8,991

Net Income: €41,009

Sample Calculation Details:
Gross Income = €50,000

USC Calculation:
First €12,012: 0.5% = €60.06
Next €15,370: 2% = €307.40
Remaining €22,618: 3% = €678.54
Total USC = €1,046.00
PRSI Calculation:
4.1% of €50,000 = €2,050.00
Income Tax Calculation:
First €44,000: 20% = €8,800
Remaining €6,000: 40% = €2,400
Total Income Tax = €11,200.00
Tax Credits Applied:
Single Person = €2,000
Employee = €2,000
Rent = €1,000
Dependent (1) = €305
Total Credits Applied = €5,305
Final Tax Liability Calculation:
(USC + PRSI + Income Tax) - Tax Credits
(1,046.00 + 2,050.00 + 11,200.00) - 4,305
= €9,991.00
Net Income Calculation: Gross Income - Final Tax Liability
= 50,000 - 8,991.00
= €41,009.00

(30 Marks)

2. Design and implement an Inventory Management System that allows users to efficiently manage product stocks. The system should perform the following operations:
    - Add a new product: System should not allow users to add the same product more than once.
    - Delete a product
    - Update product details
    - Lookup a product

On adding, deleting, updating or looking up a product, your program should let the user know if the operation was successful or not. On looking up a product return the product name, quantity and its price. If the product is not found give the user the option to add the details they are looking up. The user can perform multiple actions; they can add a new product and subsequently delete a product without having to stop and start the program until they decide to quit.

Sample:

Input	Constraints
Product Id	Mandatory, Unique
Product Name	Case Insensitive, Mandatory
Quantity	Must be >= 0
Price per unit(€)	Must be > 0
Action	'Add' or 'Update' or 'Delete' or 'Lookup' or 'Quit'

The Output should be displayed in this format

Welcome to Inventory Management System

Choose an Action: Add, Delete, Update, Lookup, Quit

Enter Action: Add

Enter Product Id: P1001
Enter Product Name: iPhone 13
Enter Quantity: 10
Enter Price per unit(€): 699
Product Added Successfully.

Enter action: Lookup

Search by (ID/Name)? Name
Enter Product Name: Air Pods
Product not found. Would you like to add it? (Yes/No): Yes

Enter Product ID: P1002
Enter Quantity: 30
Enter Price per unit (€): 299
Product Added Successfully.

(15 Marks)

3. Create a program that calculates the Body Mass Index (BMI) of a person based on their weight and height. The program should provide unit conversion and should allow user to enter the weight in pounds, height in inches and convert them to kg and meters before calculating BMI. It should also categorize the BMI according to standard classification ranges. Formula to calculate BMI = Weight(kg)/ Height(m)$^{2.}$ It must display the calculated BMI with two decimal places and its category.

BMI Range(kg/m^2)	Category
Less than 18.5	Underweight
18.5 - 25	Normal
25 - 30	Overweight
30 – above	Obese

Sample:
Input: Enter your weight(kg): 70
Enter your height(m): 1.75
Output: BMI = 22.86 kg/m2 (Normal)

(15Marks)

4. Write a program to reverse the string without reversing its individual words. Words are separated by spaces. The string may contain leading or trailing spaces, as well as multiple spaces between words. The program should return string that only have a single space separating the words, and no extra spaces. Ensure that punctuation marks are kept in the correct positions after reversing the words. It should also result the length of each word along with the word itself in the reversed order. If the string is empty, return an empty string.

   Sample:
   Input: " I am reversing this string. "
   Output:
   Length of each word: [('string.', 7), ('this', 4), ('reversing', 9), ('am', 2), ('I', 1)]
   Reversed String: "string. this reversing am I"

   (10 Marks)

5. Using python code and a web API gather stock price data on the current stock price of Tesla, Inc. (TSLA) from yahoo finance. If the price is over 100 dollars buy, otherwise don't.

   (30 Marks)

Grading Criteria

Students must properly apply Harvard referencing style while citing all sources such as web pages, forums, research papers, and other materials.

Scale	Grade Band	Coding	API's
Exceptional	**90-100%**	•Highly efficient, clean, and well-structured code, far exceeding expected standards. •Fully implements all required features and goes beyond by including advanced features or optimizations. •Comprehensive handling of edge cases and errors, with robust debugging and logging mechanisms. •Exceptionally well-documented with clear comments, docstrings. •Excellent consistency and clarity in structure and organization of code. •Demonstrates creativity and unique problem-solving approaches, incorporating advanced Python techniques (e.g., decorators, generators).	Skillful integration and use of APIs, demonstrating deep understanding and usage of advanced API features.
Outstanding	**80-89%**	•Very clean, well-structured code that exceeds most expectations. •Implements all required features with some enhancements or extra functionality. •Handles a wide range of errors and edge cases with appropriate exception management. •Well-documented with relevant comments and explanations throughout the code. •Consistent structure and organization, with only minor issues. •Shows thoughtful approaches to solving the problem, including efficient algorithms or use of Python libraries.	Effective and efficient use of APIs, with clear understanding and some use of advanced features.

Excellent	**70-79%**	•Clean and well-organized code that exceeds the threshold in most areas. •All required features are implemented and working as expected, with minor additional elements. •Error handling is present but may not cover all edge cases. •Good use of comments and explanations where necessary. •Mostly consistent and organized, with minor issues in structure. •Demonstrates good problem-solving skills and effective use of Python features.	Good understanding of APIs, using them appropriately, though limited in advanced usage.
Very Good	**60-69%**	•Generally clean and readable code though could be improved in some areas. •Most required features are implemented correctly, with minimal issues or bugs. •Some error handling is present but might miss significant edge cases. •Adequately documented but could benefit from more detailed explanations in complex parts of the code. •Clear and understandable, though inconsistencies in structure or formatting may exist. •Solves the problem in a standard way, with a few attempts at optimization or creative approaches.	Basic usage of APIs, possibly limited to simple or well-known functions.

Good	**50-59%**	•Code is functional but may be messy or difficult to follow. •Core features are implemented, but some may not work as expected or are incomplete. •Limited error handling with basic exceptions managed; some edge cases are unhandled. •Minimal documentation, with comments only in key areas or lacking clarity. •Inconsistent or unclear structure and formatting, impacting readability. •Solves the problem adequately but without any notable optimizations or creativity.	Basic usage of APIs, possibly limited to simple or well-known functions.
Sufficient	**40-49%**	•Code is somewhat difficult to follow, with poor organization. •Basic features are implemented, but several may not function properly or are missing. •Little to no error handling, or improper management of exceptions. •Very little documentation or comments; difficult to understand the logic behind the code. •Significant issues with structure, making the code hard to read or understand. •Minimal effort to optimize or enhance the solution beyond basic functionality.	Limited or incorrect use of APIs, with only basic functionality covered.

Somewhat fulfilled	**30-39%**	•Code is disorganized and hard to understand, with major readability issues. •Only a few core features are implemented, and many are broken or incomplete. •Almost no error handling, leading to frequent crashes or incorrect output. •Lacking or completely absent; very little explanation for the code's behavior. •Disorganized and lacking structure, making the code unreadable. •Basic and uncreative approach to the task, often missing key requirements	Attempts to use APIs, but with significant errors or misapplication.
Limited fulfilment	**20-29%**	•Poorly structured code that may not even run successfully. •A few features are attempted but largely do not work. •No error handling present. •No documentation or comments. •Disorganized and non-compliant with basic structural expectations. •Barely attempts the task, lacking in effort or understanding.	Minimal to no API usage, with major errors or lack of understanding.
Unfulfilled	**0-19%**	•Code does not compile or run or is completely non-functional. •No meaningful features are implemented. •No error handling at all. •No documentation or comments whatsoever. •Complete disregard for code structure or practices. •Shows no attempt at solving the problem.	No attempt to use APIs.
MIMLO Alignment (see Module Guide for Minimum Intended Learning Outcomes for Module)			
		2,4	5

General Requirements for Students:

1. All assignments must be submitted no later than the stated deadline (16/03/2025 , 23:59).
2. Assignments submitted after the latest deadline specified (including any approved extension deadline) are considered late and penalized according to the Quality Assurance Handbook (QAH)

Part B Section 5.2.2.6 as follows:

a. A penalty of 2 marks will be applied per day or part thereof (including weekends and public holidays) for an ongoing failure to submit beyond the submission deadline.
b. An examiner has the right to refuse to mark the assignment if the submission instructions have not been observed.
c. Where a late assessment is submitted within 14 days of the deadline, and is of a passing standard, the late penalty is capped (such that the minimum grade that can be awarded is 40% for the late submission).
d. Where a late assessment is submitted more than 14 days after the deadline, it will receive 0%. The lecturer may, at their discretion, review the submission for feedback.
e. Where the assessment is undertaken in a group, the piece of work should be submitted in its complete entirety, and any penalty for late submission incurred applies to all group members.

3. Extensions to assignment submission deadlines will not be granted, other than in exceptional circumstances. To apply for an extension please go to https://students.dbs.ie/dashboard/SCCM and open a ticket.
4. All relevant provisions of the Assessment Regulations must be complied with (see QAH B.5).
   a. Students are required to refer to the assessment regulations in their Programme Handbook, and on the Student Website.
   b. Dublin Business School penalises students who engage in academic impropriety (i.e. plagiarism, collusion and/or copying, ghost writing/ essay mills, improper use of Generative Artificial Intelligence software).
      i. Refer to the College's Generative AI Guidelines HERE for further information.
   c. Guides on referencing are available on the Library website: https://libguides.dbs.ie/referencing
   d. Text-matching analysis software is integrated in Moodle to generate a report regarding the degree of text-matching in a submission.
5. Students are required to retain a copy of each assignment submitted, until the issuing of a transcript indicating the mark awarded and the closure of the Appeal period (2 weeks following the release of final results).
   a. Results can only be appealed following the release of final results, and the Appeal form must be submitted to the Exams Office within the Appeal period.
   b. An appeal must be based on valid grounds (see the Appeals Policy QAH B.3.5), dissatisfaction with a grade is not sufficient grounds for an appeal.
   c. Assignments must be appropriately packaged and presented.
   d. All assignments should be submitted to your subject/course page on Moodle by the deadline date.
   e. Where a submission involves digital media (i.e formats other than Word, Powerpoint or PDF), it is the submitting students' responsibility to ensure the media is appropriately labelled, fully working and they must retain a copy[2].

f. Components of an assessment which are not included in the final submission cannot normally be subsequently accepted for grading. It is the student's responsibility to ensure their file is uploaded correctly.

6. Assignments that *breach* the word count requirements will be penalised. *There is a 10% discretion, either way, applicable in terms of word count.*

7. When you submit your assignment you will be asked to click on a button which will declare the following:

By ticking this box I am confirming that this assignment/exam is all my own work. Any sources used have been referenced. *I have read the College rules regarding plagiarism in the QAH Part B Section 3 and understand that penalties will be applied accordingly if work is found not to be my own. All work uploaded is submitted via Ouriginal, whereby a text-matching report will show any similarities with other texts.*