T Level Technical Qualification in Digital Production, Design and Development

Mark Scheme (Results)

Summer 2023

Employer Set Project

General Marking Guidance

- All students must receive the same treatment. Examiners must mark the first student in exactly the same way as they mark the last.
- Mark schemes should be applied positively. Students must be rewarded for what they have shown they can do rather than penalised for omissions.
- Examiners should mark according to the mark scheme not according to their perception of where the grade boundaries may lie.
- All marks on the mark scheme should be used appropriately.
- All the marks on the mark scheme are designed to be awarded. Examiners should always award full marks if deserved. Examiners should also be prepared to award zero marks if the student's response is not rewardable according to the mark scheme.
- Where judgement is required, a mark scheme will provide the principles by which marks will be awarded.
- When examiners are in doubt regarding the application of the mark scheme to a student's response, a senior examiner should be consulted.
- Crossed out work should be marked **unless** the student has replaced it with an alternative response.
- Accept incorrect/phonetic spelling (as long as the term is recognisable) unless instructed otherwise.

Levels-Based Mark Scheme Guidance

Levels-based mark schemes (LBMS) have been designed to assess students' work holistically. They consist of two parts:

1) **Indicative content**
   Indicative content reflects content-related points that a student might make but is not an exhaustive list. Nor is it a model answer. Students may make some or none of the points included in the indicative content as its purpose is as a guide for the relevance and expectation of the responses. Students must be credited for any appropriate response.

2) **Levels-based descriptors**
   Each level is made up of a number of traits which when combined together articulate the quality of response that a student needs to demonstrate. The traits progress across the levels to demonstrate the different expectations of each level. When using a levels-based mark scheme, the 'best fit' approach should be used.

Applying the levels-based descriptors

Examiners should take a 'best fit' approach to determining the mark.

- Examiners should first make a holistic judgement on which level most closely matches the student's response. Students will be placed in the level that best describes their answer. Answers can display characteristics from more than one level, and where this happens markers must use any additional guidance (e.g. weighting of traits) and their professional judgement to decide which level is most appropriate.

- The mark awarded within the level will be decided based on the quality of the answer and will be modified according to how securely all traits are displayed at that level:

  - Marks will be awarded at the top of that level if the student has evidenced each of the descriptor traits securely.
  - Where the response does not securely meet all traits, the marks should be awarded based on how closely the descriptor has been met.

Task 1 - Planning a project

Indicative content and marker guidance

Gannt chart:

- Expect to see tasks broken down into smaller chunks where sensible, for example:
  - May show use of an Agile approach (or similar)
  - Large modules (e.g. back-end database, data analysis etc.) broken down into multiple sections of development and unit testing with logical resources being applied to tasks – look out for students applying the same developer to test the modules – can be ok if justified through testing experience.
  - When splitting tasks students should show an understanding of how total development hours should be split between the team working on it.
  - Hardware testing is a total value so at the higher end would expect this to be split between the five restaurants.
- There should be sensible use of concurrent and serial tasks, for example:
  - Logical task sequences, e.g. cloud server setup installed before module deployed, web portal may be partially developed before but not deployed to after cloud.
  - Showing that as one unit is being tested, development could be taking place with other team members.
  - Integration testing would be expected after specific modules have been unit tested.
  - At the higher end, expect to see consideration around the testing time for each module and how this could be split up to allow testing along with other modules.
  - Possible to see an agile approach on a granular level, e.g. per module as well as the project as a whole.
  - Should show some awareness of the requirements of tasks having predecessor requirements.
- There is no single correct way organise the plan, but task orders should be sensible, for example 'create a test plan' should occur BEFORE testing commences, staff training would occur much later in the process when the system is nearer completion etc.
- The order and implementation of the project may vary significantly depending on the SDLC approach that they are taking (check against student rationale), for example:
  - a RAD/agile that looks at a Minimum Value Product (MVP) they may 'Deploy' some of the modules very early on, after a short portion of development time, and then test and develop further, deploying more modules as they go.
  - Or they may decide that their approach is to have most modules mostly developed and then deployed and tested later.

The rationale will show the reasoning for the chosen project development approach demonstrated in the Gantt chart. Points students may consider, although some of these will vary depending on the choices students make in terms of organisation. These include but are not limited to:

- Staff (skills, industry expertise, experience)
  - The senior software engineer is very experienced in development and testing. Has worked on small-scale projects so this quite large project may be a challenge. Is an experienced team leader so should be able to manage their staff well.
  - The junior software engineer is quite inexperienced but has highly relevant sills for this project. May need extra time and support due to inexperience, which could slow/delay the project.
  - The web engineer is a highly experienced developer. Due to their experience as a full stack developer, when they are not working on specific web-based tasks, they could be used to support/split the work of the junior software developer. A full stack developer is good at front-end, and back-end so could be used on most tasks.
  - The hardware and networking technician has the clear expertise in the field of overseeing large-scale infrastructure upgrades so expect to see them in these tasks – if assigned elsewhere expect to see clear justifications in rationale – could be seen around the testing elements.
  - The database engineer is an experienced member of staff - as well as working on the database module, expect to see this team member testing across the range of modules. Due to their previous experience as a software engineer, they could also test software and maybe work paired with the junior software engineer.
  - Testing should be allocated to members of staff in a sensible manner – in some cases, in student work, testers are being allocated as they are the developer – in some cases this can make sense – if clearly justified in the rationale.
- Resources
  - Students should rationalise the choice of allocation of tasks to different members of staff (see above).
  - A justification of which server option they opt for.
  - Though costings vary – should show some logical methods used to calculate.

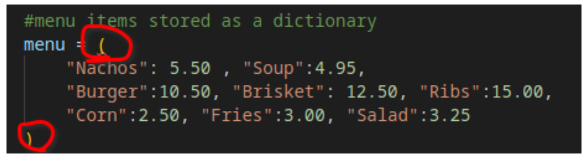
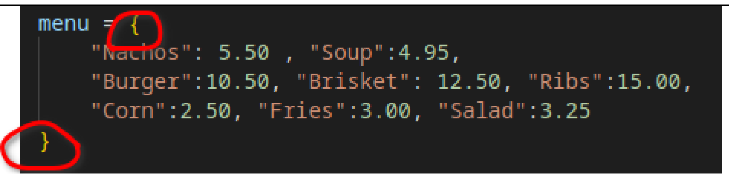
- Timescales and costs
    - Students should identify if they made the deadline or not. Using very simple calculations, the total work hours for the project is 1018. Twenty weeks with five employees provides a total of 4000 available hours to allocate. The number of hours for the project and available should be comfortable and should be able to be scheduled within the requested time.
        - Total hours for the project is simply the total work hours and does not directly relate to the number of days the project will take. There should be consideration of:
        - which jobs can only be done by specific individuals
        - splitting of jobs between suitable team members
        - contingency time.
    - More contingency time may be needed on jobs performed by the junior staff members/senior members assigned to oversee work.
    - Students should discuss the cost of their solution with reference to how these costs were arrived at, and if the projected costs and increase in revenue make the project feasible – showing awareness that the project benefits will be seen over a number of years.
    - Students could look to justify using less experienced members of staff in order to keep costings down, though mitigation (e.g. overview by the senior, more time to allow for development) and rationale should be considered.
- Risks
    - A very small team for what is quite a large project.
    - Deadlines likely to be tight.
    - Some staff have key skills but from different industries and previous roles, as this isn’t their main role this may impact on quality – quality vs meeting deadlines could be discussed by the student.
    - It is unrealistic to assume that there will be no staff absence in what will be a large-scale project - wise to build contingency time into the project as a whole and/or the tasks assigned to each member of the team – this would be for higher level students.
    - Discussion of the relevant risks of a purely cloud deployed system.
    - Upgrading multiple restaurants – delay at one could impact the scheduled upgrade of another – especially given that there is only one hardware technician – 35 hours per restaurant (upgrades and HW installation) * 5 restaurants = 175 hours total work, which is almost 22 days of work – nearly 5 weeks of the 20-week project.
    - Timescale consideration around getting the project complete on time.

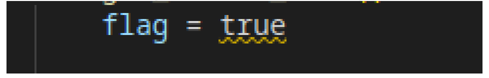
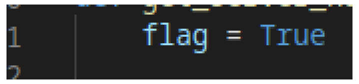
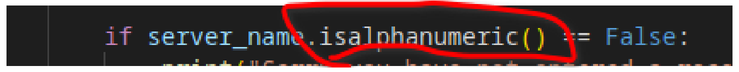
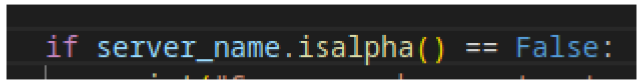
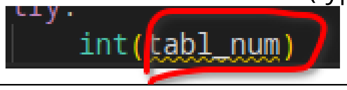
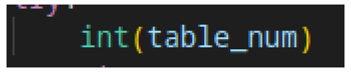
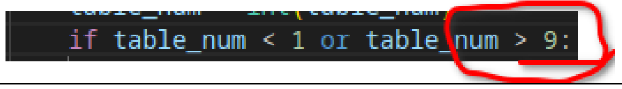
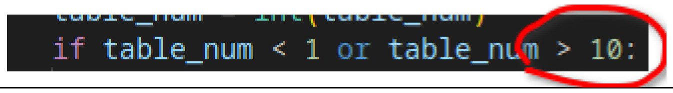
Assessment focus	Band 0	Band 1	Band 2	Band 3
	0	1-2	3-4	5-6
Gantt chart	No rewardable material	Project tasks are somewhat organised in logical and efficient manner making some use of an appropriate SDLC model to provide some accurate prediction of the project's timescales. Resources have sometimes been assigned to project tasks effectively but there are some major and/or significant errors or omissions.	Project tasks are organised in a mostly logical and efficient manner making use of an appropriate SDLC model to provide mostly accurate predictions of the project's timescales. Resources have mostly been assigned to the project tasks effectively, but there are some minor errors or omissions.	Project tasks are organised in a thoroughly logical and efficient manner using an appropriate SDLC model to provide thoroughly accurate predictions of the project's timescales. Resources have consistently been assigned to the project tasks effectively.
	0	1	2-3	4
Resource and cost plan		Some correct resources and accurate costs have been added to the plan resulting in an estimate of limited accuracy.	Mostly correct resources and accurate costs have been added to the plan resulting in an estimate that is largely accurate.	Fully correct resources and accurate costs have been added to the plan resulting in an accurate estimate.
	0	1-3	4-6	7-9
Rationale	No rewardable material	Rationale for project planning decisions demonstrate some effective consideration of: ● cost, risks and benefits ● order and timing of tasks ● selection and allocation of resources ● dependencies and prerequisites.	Rationale for project planning decisions demonstrate mostly effective consideration of: ● cost, risks and benefits ● order and timing of tasks ● selection and allocation of resources ● dependencies and prerequisites.	Rationale for project planning decisions demonstrate a thorough and perceptive consideration of: ● cost, risk and benefits ● order and timing of tasks ● selection and allocation of resources ● dependencies and prerequisites.

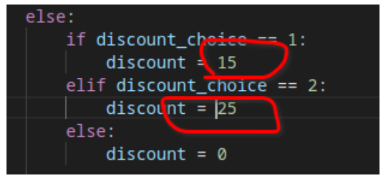
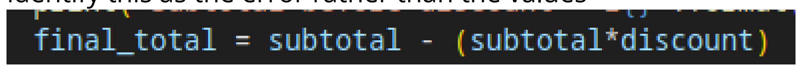
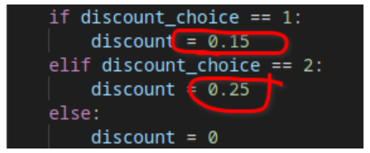
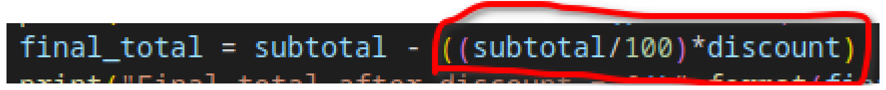
Task 2 - Identifying and fixing defects in an existing code

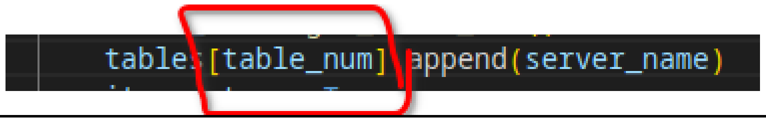
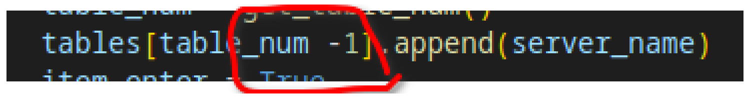
Indicative content and marker guidance

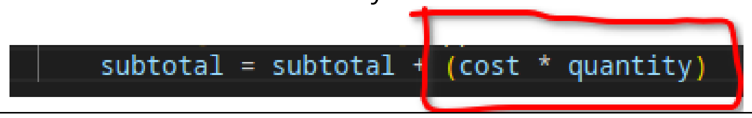
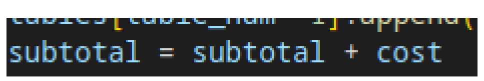
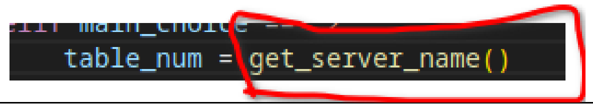
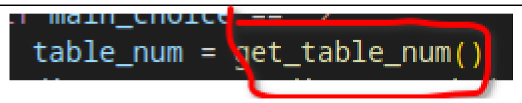
Line number	Description of error	Possible fix

4 & 8	Dictionary uses parentheses instead of curly brackets:
	
	`#menu items stored as a dictionary`
	`menu = (`
	`    "Nachos": 5.50 , "Soup":4.95,`
	`    "Burger":10.50, "Brisket": 12.50, "Ribs":15.00,`
	`    "Corn":2.50, "Fries":3.00, "Salad":3.25`
	`)`
		
		`menu = {`
		`    "Nachos": 5.50 , "Soup":4.95,`
		`    "Burger":10.50, "Brisket": 12.50, "Ribs":15.00,`
		`    "Corn":2.50, "Fries":3.00, "Salad":3.25`
		`}`

13	Incorrect capitalisation of key/reserved word ‘True’  `flag = true`	 `flag = True`
19	Incorrect string method – should be isalpha() and not isalhpanumaric()  `if server_name.isalphanumeric() == False:`	 `if server_name.isalpha() == False:`
38	Incorrect variable name (typo)  `int(tabl_num)`	 `int(table_num)`
44	Incorrect upper range for table number – should be 10  `if table_num < 1 or table_num > 9:`	 `if table_num < 1 or table_num > 10:`

126 & 128	Discount is calculated incorrectly
	
	`else:`
	`    if discount_choice == 1:`
	`        discount = 15`
	`    elif discount_choice == 2:`
	`        discount = 25`
	`    else:`
	`        discount = 0`
	Line 190 - This calculation is correct if values were 0.15 and 0.25 – students but leaners may identify this as the error rather than the values
	
	`final_total = subtotal - (subtotal*discount)`
		Note there are two possible fixes to this problem. Either changing the values in lines 126 & 128 OR changing the calculation on line 190
		
		`if discount_choice == 1:`
		`    discount = 0.15`
		`elif discount_choice == 2:`
		`    discount = 0.25`
		`else:`
		`    discount = 0`
		Solution if calculation changed instead of value:
		
		`final_total = subtotal - ((subtotal/100)*discount)`

156	Appends server_name to wrong list – uses incorrect index  `tables[table_num].append(server_name)`	 `tables[table_num -1].append(server_name)`

175	Subtotal calculated incorrectly  `subtotal = subtotal + (cost * quantity)`	 `subtotal = subtotal + cost`
181	Calls incorrect function  `table_num = get_server_name()`	 `table_num = get_table_num()`

The test plan/log should also contain inclusion of tests that show how a range of aspects of the program have been tested, including areas of the program that appear to be coded correctly have been tested to ensure outputs are correct and the program is robust. These may include (but not limited to):

- Using different menu items to test that the code correctly locates dictionary items and rejects ones that are not included.
- Using different values test validation of main menu and discount menu options including text and numbers outside of range. i.e. lower than 1 and greater than 3 for both discount and main menu.
- Testing table number boundary/index, e.g. upper limit using 9 10 and 11 to test logic is correct.
- Manual calculations of bills compared to program outputs to check calculations are correct.
- Checking the formatting of the outputted data.

A limited understanding of program requirements would be typified by only identifying and fixing the functional errors that would be highlighted by the IDE but would not identify and fix logical errors.

The number of errors identified is not a hurdle between Bands 2 and 3, the discriminator is the quality and appropriateness of the tests and data selected, and the level of understanding shown of the process.

Assessment focus	Band 0	Band 1	Band 2	Band 3
	0	1-2	3--5	6-8
Use of testing to identify defects	No rewardable material	Tests selected show a basic understanding of the identified program requirements. Test log includes some appropriate test data. Testing has identified some error in the code provided.	Tests selected show a good understanding of the identified program requirements. Test log includes a good a range of normal, erroneous and extreme data. Testing has identified most errors in the code provided.	Tests selected show a thorough and detailed understanding of the identified program requirements. Test log includes a comprehensive range of normal, erroneous and extreme data. Testing has comprehensively identified the errors in the code provided.
		1	2-3	4
Understanding of the testing process		Test log shows a basic understanding of how errors/problems were identified and how they were rectified.	Test log shows a good understanding of how errors/problems were identified and how they were rectified.	Test log shows a thorough and detailed understanding of how errors/problems were identified and how they were rectified.
		1-3	4-6	7-9
The solution		Code has some functionality, but significant errors still persist. Changes made apply some precise logic and programming structures which would result in some correct outcomes.	Code is mostly functional, but some minor errors still persist. Changes made apply mostly precise logic and programming structures which would result in mostly correct outcomes.	Code is fully functional. Changes to code apply fully precise logic and programming structures throughout which would result in consistently correct outcomes.

Task 3 – Designing a solution

Indicative content and marker guidance

General guidance:

- Algorithm designs should demostrate decomposition of the problem into simpler and more understood primitives.
- The design should provide high level coverage of the process as well as identify **reusable components.**
- Detailed algorithms (pseudocode) do not need to be provided for ALL repeated processes. For example, if the process for calculating sales for different menu items is very similar, the student would not need to provide algorithms for each region, rather they should show how **reusable** code would be utilised and may provide some additional annotation to explain the process as necessary.
- Good decomposition will show all the necessary processes and sub-processes that make up the main problem. These may include:
    - Importing data from the CSV file
    - Taking input to select item
    - Iterating through data for specified data
    - Taking a date range specified for data
    - A calculation related to 'popularity'
    - Generation of data frames and graphs
    - Aspects related to validating user input.

Some general characteristics of a good algorithm that may be demonstrated are:

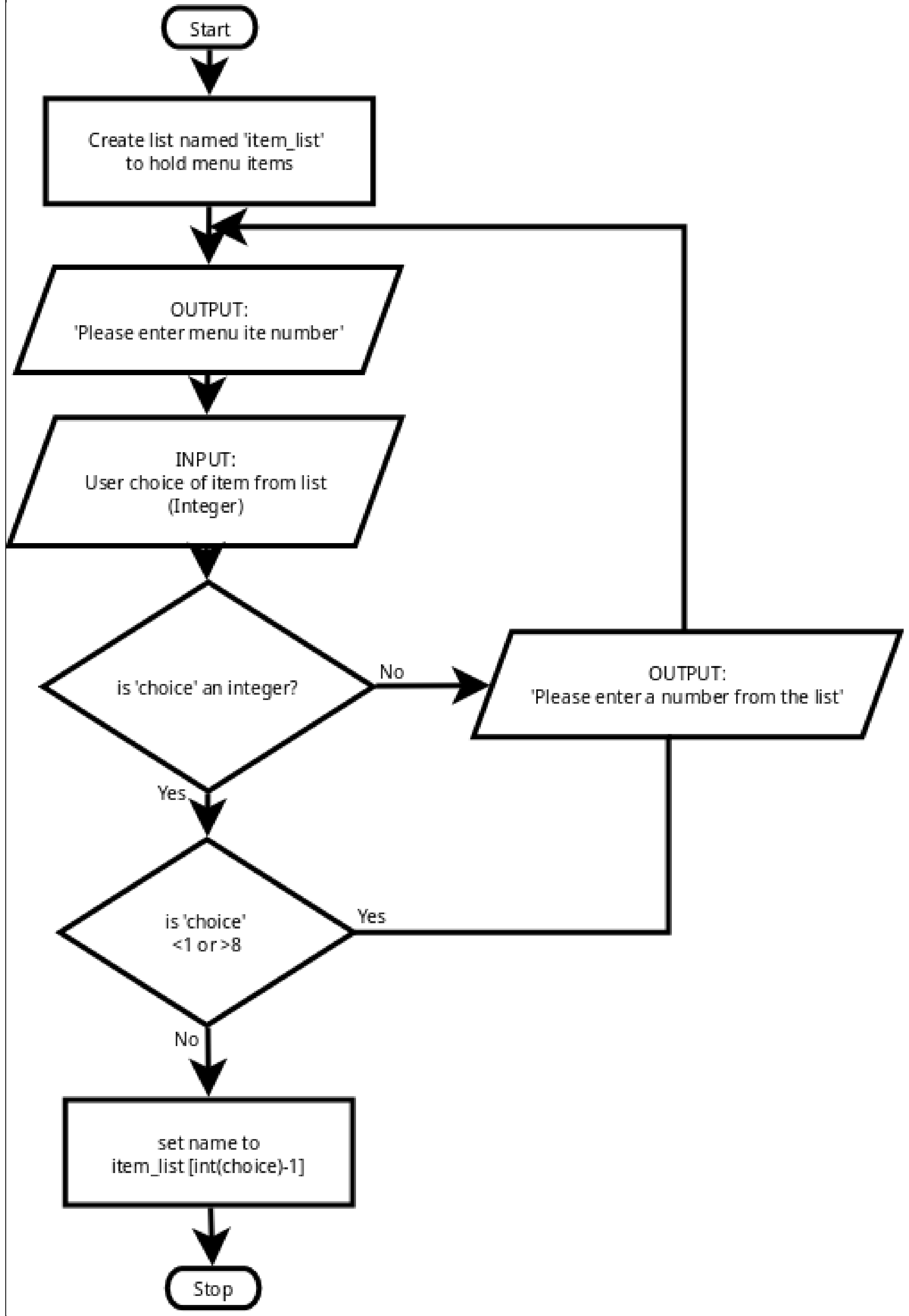
- The steps are clearly defined
- Each step is uniquely defined and should depend on the input and the result of the preceding steps
- Receives input, selection of regions and dates, as well as salespeople – inputs for user interface should also be considered
- Produces appropriate type of output, e.g. screen display, return value or return list, which results are required, what happens if no results can be computed, maybe error
- sensible naming conventions for variables and processes
- use of key words, symbols, hierarchies, and structures as appropriate to the chosen method to represent the algorithm (i.e., pseudocode or flow chart).

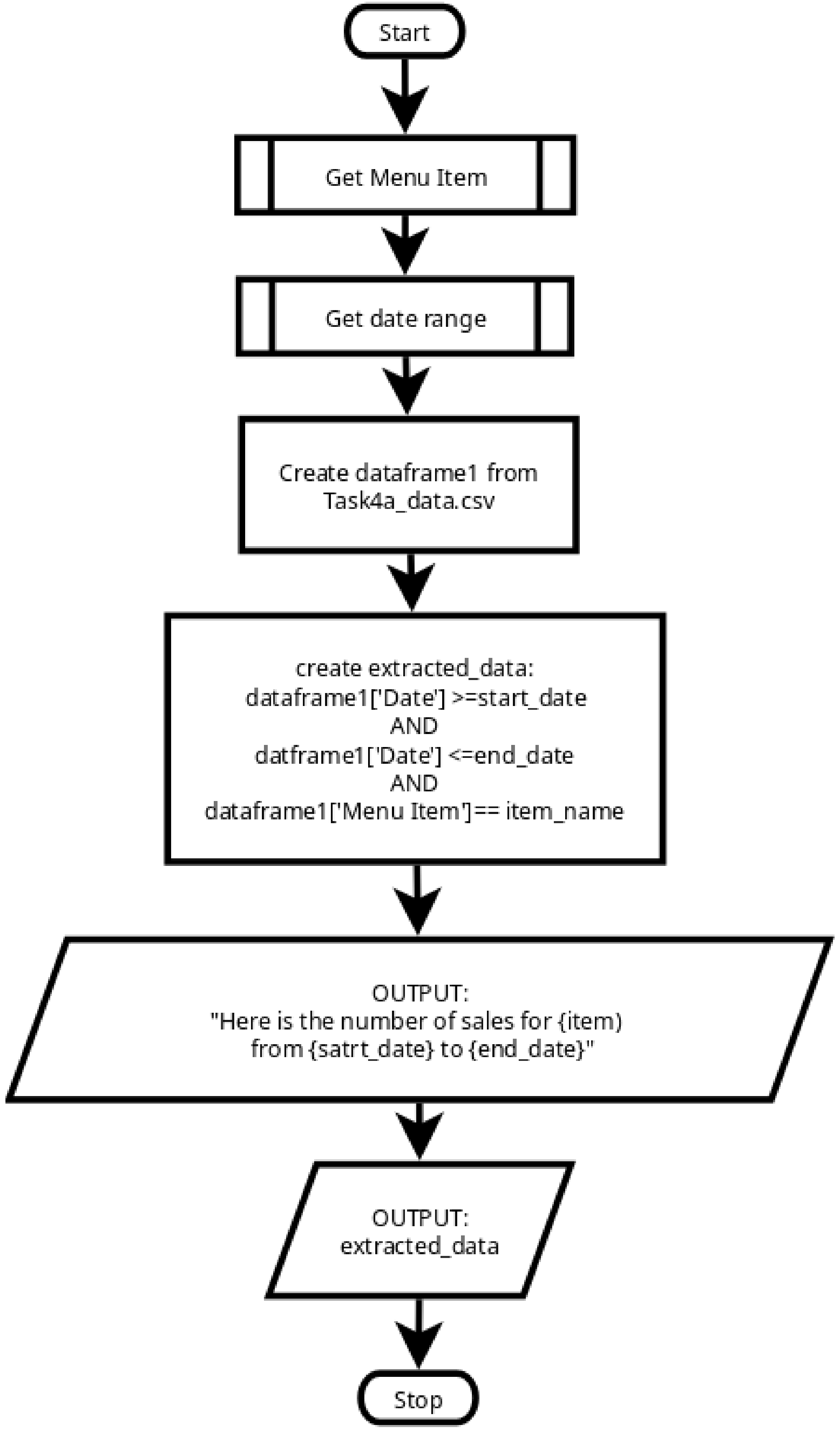
Scenario specific characteristics may include:

- Suitable logic and calculations to define date range
- Suitable calculations to calculate 'popularity', e.g. average sales per item, total sales etc.
- Links to CSV to get items and sales
- Sensible use of CSV or run-time data structure (e.g. data frame) to hold different parts of the data for processing, e.g.
    - List to hold menu options, or predefined date ranges
    - New data frame to hold data for the selected date range to aid calculations
- Understanding of given data such as:
    - Use of header row in CSV to retrieve required data (e.g. id, forename, surname)
    - Need to convert dates to a usable date format
- Simplification for user, e.g. choose a number from menu rather than type the Item in full.

Example flow chart algorithms

Get and validate Item choice


Start
Create list named 'item_list' to hold menu items
OUTPUT: 'Please enter menu ite number'
INPUT: User choice of item from list (Integer)
is 'choice' an integer?
No
OUTPUT: 'Please enter a number from the list'
Yes
is 'choice' <1 or >8
Yes
No
set name to item_list [int(choice)-1]
Stop


Start
Get Menu Item
Get date range
Create dataframe1 from Task4a_data.csv
create extracted_data: dataframe1['Date'] >=start_date AND datframe1['Date'] <=end_date AND dataframe1['Menu Item']== item_name
OUTPUT: "Here is the number of sales for {item) from {satrt_date} to {end_date}"
OUTPUT: extracted_data
Stop

Example detailed algorithms for repeated processes:

Note – these are intended to be indicative of the types of algorithm that may be presented. These **do not** show all processes. Accept any responses that provide logically correct outcomes/solutions.

Get and validate item choice

```
WHILE not_valid_input_flag = TRUE
        SEND ‘"Please enter the number of the item you wish to view TO DISPLAY

        SET item_list TO [<list of items>]

        RECEIVE item_choice (integer) FROM KEYBOARD

        IF item_choice NOT integer THEN:
                SEND ‘That is not  a valid choice’ TO DISPLAY
                SET not_valid_input_flag TO TRUE
        ELSE IF item_choice <1 OR >8 THEN:
                SEND ‘That is not  a valid choice’ TO DISPLAY
                SET not_valid_input_flag TO TRUE
        ELSE:
                SET item_choice TO item_list [int(item_choice)-1]
        RETURN item_name
```

Get and validate start date – CAST to date format

```
   SET non_valid_flag TO TRUE

   while flag == TRUE
      SEND ‘'Please enter start date for your time range DD/MM/YY') TO DISPLAY

       RECEIVE input (STRING) FROM KEYBOARD

      try:
        SET input(STRING) TO input(DATETIME)
      except:
         SEND "Sorry, you did not enter a valid date" TO DISPLAY
         flag = True
      else:
         SET start_date TO input(DATETIME)
         RETURN start_date
```

Get and display data based on date and menu item

```
 IMPORT data handling library

 READ "Task3a_data.csv"
 SET dataframe TO Task3a_data.csv

 SET results TO  datafarame.loc[( ["Date"] >= start_date) & (["Date"] <= end_date) & (["Menu Item"] == item_name)]

 SET results_print TO results.to_string(index=False)

 SEND results_print TO DISPLAY
```

Assessment focus	Band 0	Band 1	Band 2	Band 3
	0	1-2	3-5	6-8
Decomposition of the problem	No rewardable material	Basic or superficial decomposition of the identified problems that superficially cover the required: • inputs • processes • outputs.	Mostly detailed decomposition of the identified problems that sufficiently cover the required: • inputs • processes • outputs.	Thorough and detailed decomposition of the identified problems that comprehensively cover the required: • inputs • processes • outputs.
		1-2	3-4	5-6
Application of logical thinking and conventions		Algorithms would produce some correct outcomes as a result of: • some precise logic • some appropriate structure and sequence which is likely to be inefficient. Some use of accepted conventions.	Algorithms would produce mostly correct outcomes as a result of: • mostly precise logic • appropriate structure and sequence but which may lack efficiency. Mostly appropriate use of accepted conventions though some minor inconsistencies may still exist.	Algorithms would produce correct outcomes as a result of: • precise logic • appropriate and efficient structure and sequence. Appropriate and consistent use of accepted conventions.
		1	2	3
Communication of the design		Superficial communication of the design uses technical language which is only sometimes appropriate for the audience.	Adequate communication of the design uses technical language which is mostly appropriate for the audience.	Effective communication of the design uses technical language which is appropriate for the audience.

Task 4a - Developing the solution

Indicative content and marker guidance

The solution

- Provides a developed coded solution that utilises the given code and adds the additional functionality as stated in the requirements.
- Integration of existing code may include:
    - Import' function to pull code as a whole in when needed (note given code and student code will need to be in the same folder)
    - Integration into student's own code base as a function.
- The solution will be well structured and modular in nature – clearly see subsections, this may be separated modules or the use of Procedures, Functions, or classes.
- Code will be annotated to aid future maintenance of the system.
- Data/information should be output in a meaningful way to the user. This may include use of a data frame, graph and/or text-based summary.
- Output data should show consideration of different salespeople over time, but this may be interpreted in slightly different ways. Such as:
    - Higher level responses will make better use of patterns and trends over time and allow the user to select menu items, staff members and particular services in different combinations to give more meaningful data analysis
    - Lower level responses may only extract based on a single criterion, e.g. a single menu item, or calculate using a number of days, e.g. one date minus another, as opposed to a user selected date range
    - Students make use of tabulated formats or line graphs to display this information in a clearer way
    - Lower level responses may extract some meaningful information but may only output in an unsorted fashion or may not give the user sufficient choice.
- There are different ways that this task can be interpreted so the characteristics of higher level responses will show a greater discrimination of the data to be extracted.
- Code should be robust, typical errors that may be accounted for in this scenario include negative values, non-numerical characters, entering a choice not provided in the menu.

Possible areas included that contribute to 'user experience':

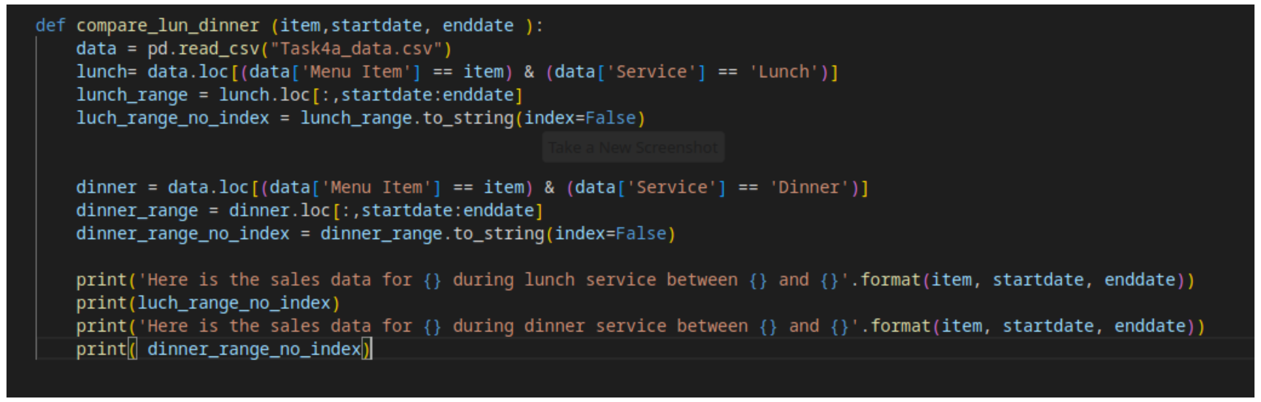
- Outputs are meaningful and make sense to the user, e.g. outputs are accompanied by meaningful text to contextualise them.
- Simplification of input processes, e.g. use of numbers in a menu rather than writing the full item names.
- User input converted from string to a usable date format.
- Potential issue of difference between input date format and date format of imported data solved (e.g. "dayfirst = True").
- Creation of new columns in a data frame to calculate subtotals and profits.
- Numerical values are rounded to a sensible number of decimal places and currency symbol added as appropriate
- Use of visualisation, e.g. bar graphs to show sales over time.
- Helpful messages and robust input handling.

Security considerations relating to the solutions could include:

- Avoiding global variables, passing data back from functions.
- Avoiding the use of a single generated data frame to ensure security of the data – good practice to generate a new data frame within a function.
- Error handling: If the system crashes, is any data returned in the error message?

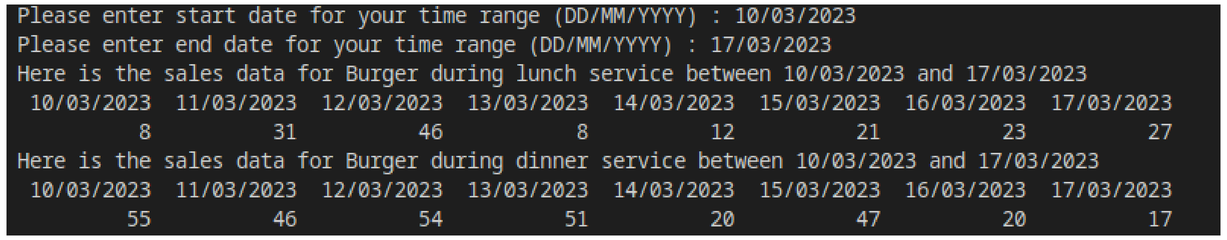
Example code snippets for parts of the solution:

Note – these are intended to be indicative of the types of response that may be presented. These **do not** show all processes. Accept any responses that provide logically correct outcomes/solutions.



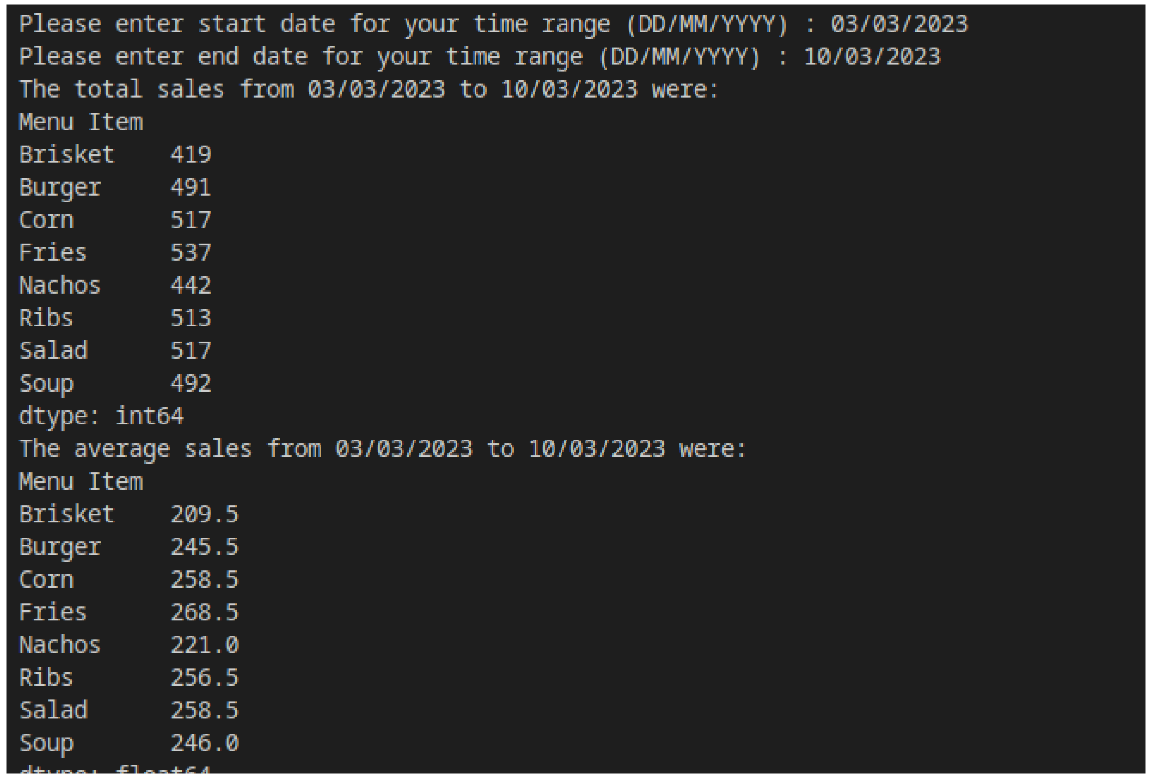
```
def compare_lun_dinner (item,startdate, enddate ):
    data = pd.read_csv("Task4a_data.csv")
    lunch= data.loc[(data['Menu Item'] == item) & (data['Service'] == 'Lunch')]
    lunch_range = lunch.loc[:,startdate:enddate]
    luch_range_no_index = lunch_range.to_string(index=False)


    dinner = data.loc[(data['Menu Item'] == item) & (data['Service'] == 'Dinner')]
    dinner_range = dinner.loc[:,startdate:enddate]
    dinner_range_no_index = dinner_range.to_string(index=False)

    print('Here is the sales data for {} during lunch service between {} and {}'.format(item, startdate, enddate))
    print(luch_range_no_index)
    print('Here is the sales data for {} during dinner service between {} and {}'.format(item, startdate, enddate))
    print( dinner_range_no_index)
```

Function to get lunch and dinner service data over a given date range



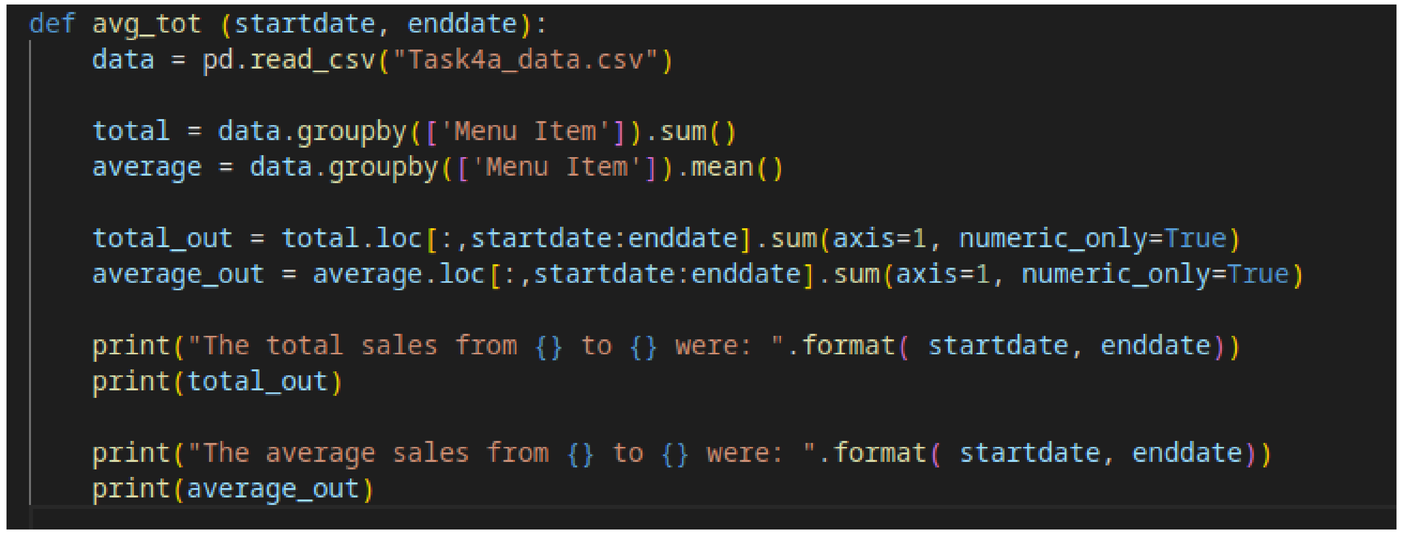
```
Please enter start date for your time range (DD/MM/YYYY) : 10/03/2023
Please enter end date for your time range (DD/MM/YYYY) : 17/03/2023
Here is the sales data for Burger during lunch service between 10/03/2023 and 17/03/2023
 10/03/2023  11/03/2023  12/03/2023  13/03/2023  14/03/2023  15/03/2023  16/03/2023  17/03/2023
          8          31          46           8          12          21          23          27
Here is the sales data for Burger during dinner service between 10/03/2023 and 17/03/2023
 10/03/2023  11/03/2023  12/03/2023  13/03/2023  14/03/2023  15/03/2023  16/03/2023  17/03/2023
         55          46          54          51          20          47          20          17
```

Showing total and average sales over a user specified range



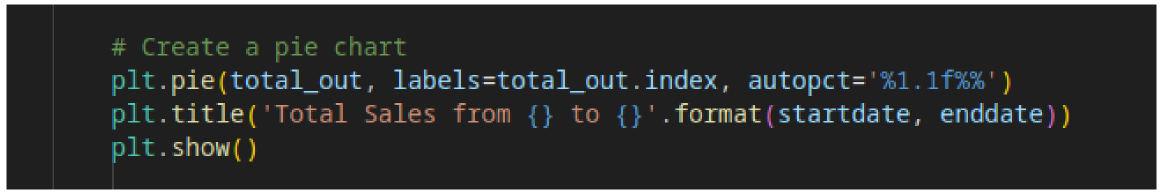
```
Please enter start date for your time range (DD/MM/YYYY) : 03/03/2023
Please enter end date for your time range (DD/MM/YYYY) : 10/03/2023
The total sales from 03/03/2023 to 10/03/2023 were:
Menu Item
Brisket    419
Burger     491
Corn       517
Fries      537
Nachos     442
Ribs       513
Salad      517
Soup       492
dtype: int64
The average sales from 03/03/2023 to 10/03/2023 were:
Menu Item
Brisket    209.5
Burger     245.5
Corn       258.5
Fries      268.5
Nachos     221.0
Ribs       256.5
Salad      258.5
Soup       246.0
dtype: float64
```

Example output:

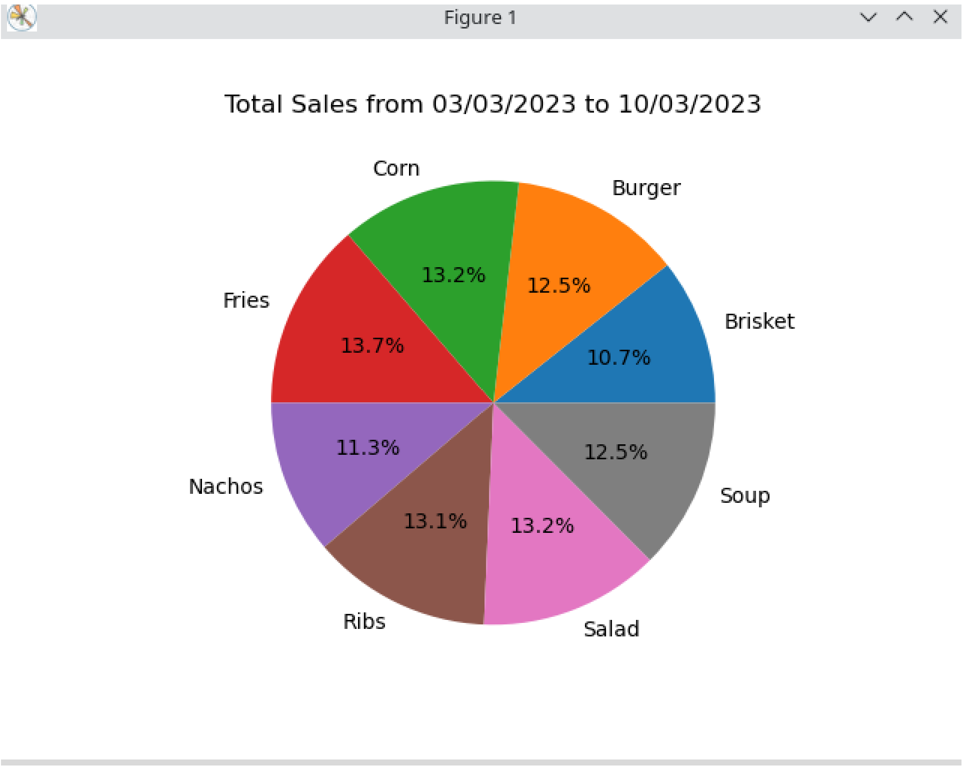


```
def avg_tot (startdate, enddate):
    data = pd.read_csv("Task4a_data.csv")

    total = data.groupby(['Menu Item']).sum()
    average = data.groupby(['Menu Item']).mean()

    total_out = total.loc[:,startdate:enddate].sum(axis=1, numeric_only=True)
    average_out = average.loc[:,startdate:enddate].sum(axis=1, numeric_only=True)

    print("The total sales from {} to {} were: ".format( startdate, enddate))
    print(total_out)

    print("The average sales from {} to {} were: ".format( startdate, enddate))
    print(average_out)
```

Use of matplotlib to produce a pie chart for total sales, from a subset of the data (to accompany the printed table)



```
# Create a pie chart
plt.pie(total_out, labels=total_out.index, autopct='%1.1f%%')
plt.title('Total Sales from {} to {}'.format(startdate, enddate))
plt.show()
```



Assessment focus	Band 0	Band 1	Band 2	Band 3	Band 4
	0	1-2	3-4	5-6	
Functionality	No rewardable material	The solution implements code with some functionality but some major errors still persist.	The solution implements mostly functional code but code may lack efficiency and some minor errors still persist.	The solution implements functional and efficient code throughout.	
		1	2	3	
Logic and programming structures		The code uses some precise logic and programming structures which would result in some correct outcomes.	The code uses mostly precise logic and programming structures which would result in sufficiently correct outcomes.	The code uses precise logic and programming structures throughout which would result in consistently correct outcomes.	
		1	2	3	
Robustness		The code handles some common user errors.	The code handles most common user errors.	The code thoroughly handles common, and most unexpected, user errors.	
		1-2	3-4	5-6	
Security		The code mitigates against some common vulnerabilities as a result of some effective application of secure coding practices.	The code mitigates against most relevant vulnerabilities through mostly effective application of secure coding practices.	The code thoroughly mitigates against relevant vulnerabilities through effective application of secure coding practices.	
		1-2	3-4	5-6	7-8
Code organisation		The code is partially maintainable by a third party but would present significant difficulties through the use of: • inconsistent naming conventions • limited logical organisation • limited informative commenting.	The code is partially maintainable by a third party but would present some minor difficulties through the use of: • some consistent naming conventions • some logical organisation • some informative commenting.	The code is maintainable by a third party and would present only a few minor difficulties through the use of: • mostly consistent naming conventions • mostly logical organisation • mostly informative commenting.	The code is easily maintainable by a third party through the use of: • consistent and appropriate naming conventions • fully logical organisation • highly informative commenting.

	1-2	3-4	5-6	7-8
User experience	Basic user experience is provided through limited effective use of: • input handling • user guidance and error messages • outputs.	Adequate user experience is provided through somewhat effective use of: • input handling • user guidance and error messages • outputs.	Good user experience is provided through mostly effective use of: • input handling • user guidance and error messages • outputs.	Excellent user experience is provided through consistently effective use of: • input handling • user guidance and error messages • outputs.

Task 4b - Reflective evaluation

Indicative content and marker guidance

Indicative content will vary according to the approach students have taken in Task 4a and the effectiveness of the solution they created.

Generic features of effective evaluations are likely to include:

- the extent to which the solution meets the:
  - requirements of the set task brief
  - needs of the users
- a justification of how the solution could be further developed/enhanced
- specific examples from the solution to support points made
- contextualisation of any points made and explaining what they did and justifying why.

Contextulisation for this scenario may include:

- How they solved issues with date formats vs strings.
- Choice of data output (e.g text, table, graph type) – which is most suitable for showing totals, averages, dinner/lunch services, data over a period of time.
- Rounding vs truncation for calculations (e.g. average sales) and consideration of data types to specify number of decimal places for the calculation output.
- How the brief was interpreted and integrated into the functionality of the program, e.g. were different services or time frames compared, was the user able to choose options from a list?
- How existing code was integrated.
- Choice of libraries/functions to get required data.
- How trends, patterns have been calculated, e.g. have user inputs for dates been considered, how has the information been grouped/sorted?
- How data was extracted (e.g. use directly form the csv file, use of subsets of data, how the choice of data frame or datafile impacted on search/extraction).
- Use of variables (global vs local, passing data between functions).
- Input error handling, e.g. why they might have excluded text or negative values, menu options etc.

Assessment focus	Band 0	Band 1	Band 2	Band 3
	0	1-2	3-4	5-6
Programming outcomes	No rewardable material	Judgements reached are somewhat supported showing a superficial or basic understanding of how well the solution met the: ● requirements of the set task brief ● needs of the users.	Judgements reached are mostly well supported showing a good understanding of how well the solution met the: ● requirements of the set task brief ● needs of the users.	Judgements reached are comprehensively well supported showing a thorough and detailed understanding of how well the solution met the: ● requirements of the set task brief ● needs of the users.
		1	2	3
Future Developments		A superficial or simplistic rationale is provided for what future developments should be implemented.	A good rationale which is reasonably well supported is provided for what future developments should be implemented.	A convincing and well-supported rationale is provided for what future developments should be implemented.