Solent University

Faculty of Business, Law and Digital Technologies

**The Software Project**

**Selecting a mobile device**

**based on PhoneDB data**

Author : **Ivan Rubinov, Q102110038 student code**

Course Title : **Programming for Problem Solving, COM728**

Module Leader : **Jarutas Andritsch**

Date **: 01.01.2024**

Table of Contents

[1. Overview 4](#_Toc155181473)

[2. Project Implementation 5](#_Toc155181474)

[2.1 Project structure 5](#_Toc155181475)

[2.2 Modules/ Functions 7](#_Toc155181476)

[2.2.1 Main module, an entry point (main\_file.ipynb) 7](#_Toc155181477)

[2.2.1.1 First part - the prompting of a file name 7](#_Toc155181478)

[2.2.1.2 The second part - the executing interface menu 7](#_Toc155181479)

[2.2.1.3 Constant with all possible interface menu options 8](#_Toc155181480)

[2.2.1.4 Constant with all possible retrieval conditions 9](#_Toc155181481)

[2.2.2 Menu\_module 10](#_Toc155181482)

[2.2.2.1 Function input\_file\_path() 10](#_Toc155181483)

[2.2.2.2 Function input\_choice() 11](#_Toc155181484)

[2.2.2.3 Functions input\_value() and input\_value\_range() 12](#_Toc155181485)

[2.2.2.4 Functions input\_feature() 13](#_Toc155181486)

[2.2.3 Retrieve\_module / function retrieve() 13](#_Toc155181487)

[2.2.3.1 Reasons for building my custom retrieval conditions 14](#_Toc155181488)

[2.2.4 Key\_match\_module / function find\_by\_key() and find\_by\_key\_range() 14](#_Toc155181489)

[2.2.5 Analytics\_module / function analytics\_visualize() 15](#_Toc155181490)

[2.2.5.1 Function avg\_group\_filter() 16](#_Toc155181491)

[2.2.5.2 Function avg\_group() 17](#_Toc155181492)

[2.2.5.3 Function count\_group\_two() and reasons to build it 18](#_Toc155181493)

[2.2.5.4 Function top\_features() 20](#_Toc155181494)

[2.2.5.5 Function pie\_chart\_filtered() 21](#_Toc155181495)

[2.2.5.6 Function chart\_counted() 23](#_Toc155181496)

[2.2.5.7 Function chart\_monthly\_price() 24](#_Toc155181497)

[2.2.5.8 Function chart\_metrics\_three() and reasons to build it 25](#_Toc155181498)

[3. GitHub Repository Evidence 27](#_Toc155181499)

Table of Figures

Figure 1 Software Project. Modules structure (adapted from Andritsch, J., 2022) 5

Figure 2 Software Project continued. Analytics module structure (adapted from Andritsch, J., 2022) 6

Figure 3 Code to call the function requesting the file name 7

Figure 4 Example of requesting the file name 7

Figure 5 Code to select a menu option and call the corresponding module 8

Figure 6 Code of possible menu options, const ‘choice\_list’ 8

Figure 7 Example of executing Interface menu 9

Figure 8 Code of possible filter conditions, const ‘retrieve\_condition’ 10

Figure 9 Code of function asking a filename, input\_file\_path() 11

Figure 10 Code of function visualising menu interface, input\_choice() 12

Figure 11 Code of functions input\_value() and input\_value\_range() 12

Figure 12 Code of function asking a feature name, input\_feature() 13

Figure 13 Beginning code of function retrieve() 14

Figure 14 Example of retrieving devices by weight 14

Figure 15 Code of functions to search by key 15

Figure 16 Beginning code of function analytics\_visualize() 16

Figure 17 Code of averaging function avg\_group\_filter() 17

Figure 18 Example of price averaging 17

Figure 19 Code of averaging function avg\_group() 18

Figure 20 Example of averaging weight for each manufacture 18

Figure 21 Code of counting function, count\_group\_two() 18

Figure 22 Code of calling analytics menu options 19

Figure 23 Example of counted devices grouped by brand and released year. 20

Figure 24 Code of counting function, top\_features() 21

Figure 25 Example of top regions for Apple brand 21

Figure 26 Code of visualising function pie\_chart\_filtered() 22

Figure 27 Example of a proportion of RAM types 22

Figure 28 Code of visualising function, chart\_counted() 23

Figure 29 Example of the number of devices for each USB connector Type 23

Figure 30 Code of visualising function, chart\_metrics\_three 24

Figure 31 Example of a monthly average price chart 24

Figure 32 Calling of visualising function, chart\_metrics\_three 25

Figure 33 Example of comparing two brands 26

Figure 34 Github commits 27

Figure 35 Commit history in GitHub 28

Index of Tables

Table 1: Requirement Completion 4

Table 2: List of all functions in alphabetical order 6

# 1. Overview

The project’s aim is a help with selecting the most suitable mobile device. Users can find and analyse devices based on features extracted from the PhoneDB website in the form of a CSV file.

This file provides technical data for smartphones, tablets, PDAs, and other mobile devices. There are technical specifications such as weight, display and dimensions, as well as marketing data such as selling prices and regions where the device is distributed.

The file contains 48 columns separated by commas. Each row represents one device released in 2020-2023 years. Possible column datatypes are text, numbers or datetime. Several columns such as software\_extracts, cpu, supported\_cellular\_bands, WLAN, markets\_regions contain multiple values within the same column.

The project delivers a proper value to users saving their time and money. Users can make better decisions about selecting a mobile device, and reveal existing trends and patterns through data analysis and visualization.

Below is a table with the requirements achieved.

Table 1: Requirement Completion

|  |  |  |
| --- | --- | --- |
| **#** | **Requirement** | **Status** |
| a | Load the data from a CSV file specified by the user using the CSV reader function. | Completed |
| a1 | Retrieve the model name, manufacturer, weight, price, and price currency based on the oem\_id. | Completed |
| a2 | Retrieve the brand, model name, RAM capacity, market regions, and the date based on the code name. | Completed |
| a3 | Retrieve the oem\_id, release date, announcement date, dimensions, and device category based on RAM capacity. | Completed |
| a4 | Custom retrieval. Three columns: hardware designer, display diagonal, sim card slot after a specific condition: weight. | Completed |
| b | Load data from a CSV file using the pandas module. | Completed |
| b1 | Identify the top 5 regions where a brand was sold. | Completed |
| b2 | Analyse the average price of a brand, all in the same currency. | Completed |
| b3 | Analyse the average mass for each manufacturer. | Completed |
| b4 | Analyse the data to derive meaningful insights based on unique selection, distinct from the previous requirements (custom). | Completed |
| c | Load data from a CSV file into memory. | Completed |
| c1 | Create a chart to represent the proportion of RAM types. | Completed |
| c2 | Create a chart with the number of devices for USB connector types. | Completed |
| c3 | Create charts illustrating the monthly average price trends (in GBP) for devices released in each year from 2020 to 2023. | Completed |
| c4 | Custom visualisation. Charts comparing two brands by 3 metrics: median memory, median weight and max selling price in USD. | Completed |

**Status options:** Completed/ Partially Completed/ Not Attempted

# 2. Project Implementation

An initial dataset is loaded in memory by using CSV reader module (*‘retrieve\_module.py’*) or pandas module (*‘analytics\_module.py’*). The path of the dataset is provided by the user. CSV reader module is used for retrieving device columns, while pandas module is used for analytics such as finding top regions or average prices. Matplotlib module is used to visualize the data and reveal trends and patterns.

#### 2.1 Project structure

The diagram below illustrates the project structure. There are five modules (files):

* Main module to run the program, an entry point
* Menu module to interact with a user and prompt input values.
* Retrieve module to find/filter devices by features
* Module to work with keys and dictionaries
* Analytics module to aggregated and visualise data



Figure 1 Software Project. Modules structure (adapted from Andritsch, J., 2022)



Figure 2 Software Project continued. Analytics module structure (adapted from Andritsch, J., 2022)

Table 2: List of all functions in alphabetical order

|  |  |  |  |
| --- | --- | --- | --- |
| **Function name** | **Module where it is defined** | **Module where it is used (called)** | **# specific task** |
| analytics\_visualize() | analytics\_module | Entry point(main module) | c |
| avg\_group() | analytics\_module | analytics\_module | b3 |
| avg\_group\_filter() | analytics\_module | analytics\_module | b2 |
| chart\_counted() | analytics\_module | analytics\_module | c2 |
| chart\_metrics\_three() | analytics\_module | analytics\_module | c4 |
| chart\_monthly\_price() | analytics\_module | analytics\_module | c3 |
| count\_group\_two() | analytics\_module | analytics\_module | b4 |
| find\_by\_key() | key\_match\_module | retrieve\_module |  |
| find\_by\_key\_range() | key\_match\_module | retrieve\_module |  |
| input\_choice() | menu\_module | Entry point(main module) |  |
| input\_feature() | menu\_module | analytics\_module |  |
| input\_file\_path() | menu\_module | Entry point(main module) |  |
| input\_value() | menu\_module | menu\_module, retrieve\_module |  |
| input\_value\_range() | menu\_module | retrieve\_module |  |
| pie\_chart\_filtered() | analytics\_module | analytics\_module | c1 |
| retrieve() | retrieve\_module | Entry point(main module) | a, a1, a2, a3, a4 |
| top\_features() | analytics\_module | analytics\_module | b1 |

#### 2.2 Modules/ Functions

##### 2.2.1 Main module, an entry point (main\_file.ipynb)

This module is an entry point to run the program and call other modules.

It consists of two parts.

###### 2.2.1.1 First part - the prompting of a file name

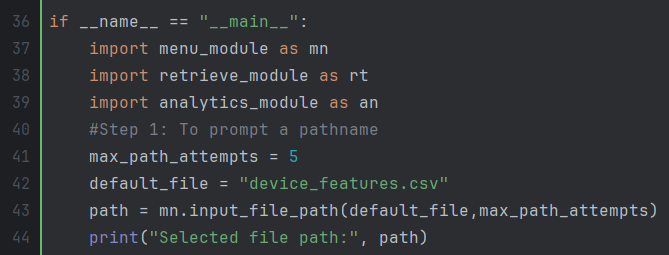


Figure 3 Code to call the function requesting the file name

Constants

* *‘max\_path\_attempts’* – the max number of attempts to request a file name
* *‘default\_file’* – the file name by default.

A screen shot of a computer code

Description automatically generated

Figure 4 Example of requesting the file name

###### 2.2.1.2 The second part - the executing interface menu

A screen shot of a computer program

Description automatically generated

Figure 5 Code to select a menu option and call the corresponding module

Variables (‘*min\_retrieve’, ‘max\_retrieve’, ‘min\_analytics’, ‘max\_analytics’*) are lower/upper boundaries to determine which module to execute: retrieve or analyse.

###### 2.2.1.3 Constant with all possible interface menu options

The value of boundaries is corresponded to index values of constant ‘*choice\_list’.* Having all the menu options in one list makes easy to add new options.

A screenshot of a computer program

Description automatically generated

Figure 6 Code of possible menu options, const ‘choice\_list’

A screenshot of a computer program

Description automatically generated

Figure 7 Example of executing Interface menu

###### 2.2.1.4 Constant with all possible retrieval conditions

A similar solution where all data is kept within one dictionary was applied to call the retrieval module.

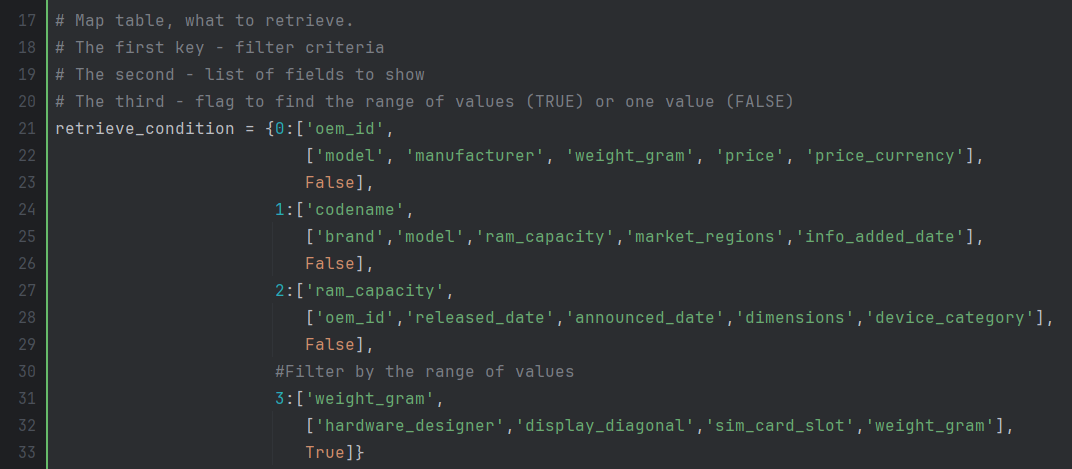


Figure 8 Code of possible filter conditions, const ‘retrieve\_condition’

In the code above there are four conditions with numbers from 0 to 3. Each condition has a list with three keys.

The first two keys are the name(s) of the device features:

* filter the dataset (like *‘oem\_id’*)
* display the results after the filtering (like *‘model’, ‘manufacturer’*)

The third key is a special flag responsible for

* searching by one value
* searching by value range.

The initial project requirements do not have such a flag, but it was helpful for finding devices by a price range in my custom search.

##### 2.2.2 Menu\_module

###### 2.2.2.1 Function input\_file\_path()

Takes two parameters:

* default path (returns it if the user leaves the prompt blank)
* max number of tries to request the file name.

Returns:

* the path

A computer screen shot of a code

Description automatically generated

Figure 9 Code of function asking a filename, input\_file\_path()

An example of usage can be found in Figure 4

###### 2.2.2.2 Function input\_choice()

Takes one parameter:

* all the possible menu options as a list

and visualises them in the menu loop.

Returns:

* the index number of the selected menu option.

A computer screen with many colorful text

Description automatically generated with medium confidence

Figure 10 Code of function visualising menu interface, input\_choice()

The example of usage can be found in Figure 7.

###### 2.2.2.3 Functions input\_value() and input\_value\_range()

Both functions take one parameter:

* Name of key

Then prompt and return:

* A value (or a value range)

A screen shot of a computer program

Description automatically generated

Figure 11 Code of functions input\_value() and input\_value\_range()

###### 2.2.2.4 Functions input\_feature()

Takes two parameters:

* the dataset to filter
* the feature name to be filtered.

Then prints all possible values and asks the user to enter the right one.

Returns:

* a value selected by the user

A screen shot of a computer code

Description automatically generated

Figure 12 Code of function asking a feature name, input\_feature()

##### 2.2.3 Retrieve\_module / function retrieve()

Takes two parameters:

* the file path
* the condition of
  + how to retrieve
  + the features to display

Displays:

* devices from the file filtered by the condition.

A screen shot of a computer program

Description automatically generated

Figure 13 Beginning code of function retrieve()

I opted to keep all conditions/feature names in one dictionary as more flexible and elegant. It would be easier to add the new conditions/feature names.

###### 2.2.3.1 Reasons for building my custom retrieval conditions

I opted to find devices by their weight because it is one of the most important technical specifications.

A screenshot of a computer program

Description automatically generated

Figure 14 Example of retrieving devices by weight

It is easy to verify that I selected three columns and one condition that differs from previous requirements if we look at Figure 8 where all columns and conditions are placed.

##### 2.2.4 Key\_match\_module / function find\_by\_key() and find\_by\_key\_range()

Both functions take:

* dataset
* values for features
* feature names to filter the dataset

Returns

* dataset filtered by values of the specified key.

A screenshot of a computer program

Description automatically generated

Figure 15 Code of functions to search by key

##### 2.2.5 Analytics\_module / function analytics\_visualize()

Takes:

* the filename
* one menu option
* font size, width and height for charts

Returns

* the values or charts based on the specified menu option.

A screen shot of a computer program

Description automatically generated

Figure 16 Beginning code of function analytics\_visualize()

The function can be easily expanded by adding more values in the case block.

There is a pre-processing of data:

* ‘*market\_regions’* is converted to the list datatype for better processing
* two additional fields are added (‘*released\_year*’ and ‘*release\_month’*).

Some examples of usage are provided below

###### 2.2.5.1 Function avg\_group\_filter()

Takes five parameters:

* dataset
* three feature names
  + to filter the dataset
  + to group the dataset
  + to average the dataset
* the output text.

Requests for

* a feature name to be filtered

Prints

* the list of average values

A screen shot of a computer program

Description automatically generated

Figure 17 Code of averaging function avg\_group\_filter()

Values formatted with two decimal values

A screenshot of a computer

Description automatically generated

Figure 18 Example of price averaging

###### 2.2.5.2 Function avg\_group()

Takes four parameters:

* Dataset
* two feature names
  + to group dataset
  + to average dataset
* output text

Prints

* the list of average values

A screen shot of a computer

Description automatically generated

Figure 19 Code of averaging function avg\_group()

Example of usage

A screen shot of a computer program

Description automatically generated

Figure 20 Example of averaging weight for each manufacture

###### 2.2.5.3 Function count\_group\_two() and reasons to build it

Takes four parameters

* Dataset
* two feature names
  + to group
  + and count results
* output text

Prints

* the numbers of rows counted and grouped by two features

A screen shot of a computer code

Description automatically generated

Figure 21 Code of counting function, count\_group\_two()

I opted to count the number of devices released grouped by *‘hardware\_designer’* because it reveals the trends the trends how the companies designed their products. We can see that Motorola and Apple was quite active before 2022 but have no products in 2023. It can be a warning sign about their activities in future or they are preparing a major update.

To demonstrate that my selection was unique and distinct from the prior requirements, the following code could be utilized. My custom selection is in case 7.

A screenshot of a computer program

Description automatically generated

Figure 22 Code of calling analytics menu options

Example of usage

A screenshot of a computer screen

Description automatically generated

Figure 23 Example of counted devices grouped by brand and released year.

###### 2.2.5.4 Function top\_features()

Takes five parameters

* Dataset
* two feature names
  + to filter the dataset
  + to group the dataset
* max number of rows to print
* output text

The function also prompts to enter the value to be filtered.

Prints

* a list of top values. The number of rows defined by the max number.

The function divides (parses) the string into substrings and counts substrings separately using the built-in function *explode()*

A screenshot of a computer program

Description automatically generated

Figure 24 Code of counting function, top\_features()

Example of usage

A screenshot of a computer

Description automatically generated

Figure 25 Example of top regions for Apple brand

###### 2.2.5.5 Function pie\_chart\_filtered()

Takes seven parameters:

* Dataset
* two feature names
  + to filter the dataset
  + to group the dataset
* fontsize, width, height, title for chart

Requests for

* a value to filter

A screen shot of a computer code

Description automatically generated

Figure 26 Code of visualising function pie\_chart\_filtered()

Visualises a pie chart with proportions.

A diagram of different types of ram types

Description automatically generated

Figure 27 Example of a proportion of RAM types

###### 2.2.5.6 Function chart\_counted()

Takes seven parameters:

* Dataset
* feature name to group the dataset
* parameters to display a chart (fontsize etc).

A screen shot of a computer program

Description automatically generated

Figure 28 Code of visualising function, chart\_counted()

Visualises a bar chart.

A graph of devices with numbers and symbols

Description automatically generated with medium confidence

Figure 29 Example of the number of devices for each USB connector Type

###### 2.2.5.7 Function chart\_monthly\_price()

Takes 14 parameters, look at names and usage in the Figure below

A screenshot of a computer program

Description automatically generated

Figure 30 Code of visualising function, chart\_metrics\_three

Visualises several annual charts, each for a specific year

A graph with blue lines and numbers

Description automatically generated

Figure 31 Example of a monthly average price chart

###### 2.2.5.8 Function chart\_metrics\_three() and reasons to build it

Takes 21 parameters, look at names in the Figure below

A screen shot of a computer program

Description automatically generated

Figure 32 Calling of visualising function, chart\_metrics\_three

Visualises three metrics of two brands

Example of usage

A graph of different brands

Description automatically generated

Figure 33 Example of comparing two brands

I decided to display this information because it reveals the existing trends for brands and what you can expect from them. In my opinion, Samsung unsuccessfully tried to move to luxury goods in 2020-2021 as can be seen from the chart with the max prices. The median weight for both brands is relatively stable, about 200 grams and I don’t expect any improvements in the next years. It is an interesting trend with median memory capacity. It looks like the new standard capacity is about 250 GB while Apple set this trend a year before Samsung.

# 3. GitHub Repository Evidence

The screenshot of my Git repository shows a history of my 35 commits of project implementation starting 10th November 2023.

A screenshot of a computer

Description automatically generated

Figure 34 Github commits

Sample screenshot of my commit history:

A screenshot of a computer

Description automatically generated

Figure 35 Commit history in GitHub