Solent University

Faculty of Business, Law and Digital Technologies

**The Software Project**

**Making a phone choice**

**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](#_Toc153723961)

[2. Project Implementation 5](#_Toc153723962)

[2.1 Project structure 5](#_Toc153723963)

[2.2 Modules/ Functions 6](#_Toc153723964)

[2.2.1 Main module, an entry point (main\_file.ipynb) 6](#_Toc153723965)

[2.2.2 Menu\_module 9](#_Toc153723966)

[2.2.2.1 Function input\_file\_path() 9](#_Toc153723967)

[2.2.2.2 Function input\_choice() 10](#_Toc153723968)

[2.2.2.3 Functions input\_value() and input\_value\_range() 11](#_Toc153723969)

[2.2.3 Retrieve\_module / function retrieve() 11](#_Toc153723970)

[2.2.4 Key\_match\_module / function find\_by\_key() and find\_by\_key\_range() 11](#_Toc153723971)

[2.2.5 Analytics\_module / function analytics\_visualize() 12](#_Toc153723972)

[2.2.5.1 Function avg\_group\_two() 13](#_Toc153723973)

[2.2.5.2 Function avg\_group\_one() 14](#_Toc153723974)

[2.2.5.3 Function count\_annualy() 15](#_Toc153723975)

[2.2.5.4 Function top\_features() 15](#_Toc153723976)

[3. GitHub Repository Evidence 18](#_Toc153723977)

Table of Figures

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

Figure 2 Code to call the function for a file name 6

Figure 3 Example of requesting a file name 6

Figure 4 Code to select for a menu option and run the corresponding module 7

Figure 5 Code of possible menu options, const ‘choice\_list’ 7

Figure 6 Example of running the menu 8

Figure 7 Code of possible filter conditions, const retrieve\_condition 9

Figure 8 Code of function asking a filename, input\_file\_path() 10

Figure 9 Code of function visualising menu interface, input\_choice() 10

Figure 10 function input\_value() and function input\_value\_range() 11

Figure 11 Beginning code of function retrieve() 11

Figure 12 Code of functions to search by key, find\_by\_key() and find\_by\_key\_range() 12

Figure 13 Beginning code of function analytics\_visualize() 12

Figure 14 Example of charts 13

Figure 15 Code of averaging function avg\_group\_two() 14

Figure 16 Example of analyse of the average price of device 14

Figure 17 Code of averaging function avg\_group\_one() 14

Figure 18 Example of averaging weight for each manufacture 15

Figure 19 Code of counting function, count\_annualy() 15

Figure 20 Example of counting device grouped by brand and released year. 15

Figure 21 Code of counting function, top\_features() 16

Figure 22 Example of top regions for Apple brand 16

Figure 23 Github commits 18

Figure 24 Commit history in GitHub 19

Index of Tables

Table 1: Requirement Completion 4

# 1. Overview

The aim of the project to help users in finding the most suitable mobile device. The project allows to find and analyse device features extracted from the PhoneDB website in the form of a CSV file. PhoneDB website provides technical data for smartphones, tablets, PDAs, and mobile.

The input text file contains 48 columns. Each row in the file represents a device released in 2020-2022 years. Data is represented as text (string datatype), numbers (xxx datype) and date (xxx datatype) divided by comma. The file has technical specifications such as weight, display or dimensions, as well as marketing data such as selling prices in different currencies and regions where device is distributed.

The extracting of data from the PhoneDB, analysing it and visualization allows users to save time, money and make the better decisions during revealing existing trends and patterns.

Below is a table summarising what requirement have been achieved.

Table 1: Requirement Completion

|  |  |
| --- | --- |
| **Requirement** | **Status** |
| Load the data from a CSV file specified by the user using the csv reader function. | Completed |
| Retrieve the model name, manufacturer, weight, price, and price currency based on the oem\_id. | Completed |
| Retrieve the brand, model name, RAM capacity, market regions, and the date based on code name. | Completed |
| Retrieve the oem\_id, release date, announcement date, dimensions, and device category based on RAM capacity. | Completed |
| Load data from a CSV file using the pandas module. | Completed |
| Identify the top 5 regions where a brand was sold. | Completed |
| Analyse the average price a brand, all in the same currency. | Completed |
| Analyse the average mass for each manufacturer | Completed |
| Create a chart to represent the proportion of RAM types | Completed |
| Create a chart to compare the number of devices for USB connector types | Completed |
| Create charts illustrating the monthly average price trends (in GBP) for devices released in each year from 2020 to 2023. | Completed |
| Create a visualisation of your selection to showcase information related to device | Completed |

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

# 2. Project Implementation

An initial dataset is loaded in memory by using of CSV reader or pandas module. The path of the dataset is provided by the user. CSV reader module is used for retrieving device names, while pandas module is used for analytics such as finding of 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

* Main module to run the program, an entry point
* Menu module is used to interact with a user.
* Retrieve module is used to filter devices by features
  + Supportive module to work with keys and dictionaries
* Analytics module is used to summarise and visualise data aggregated by

|  |
| --- |
|  |
|  |

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

#### 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.

First part - prompting a file name

A screen shot of a computer code

Description automatically generated

Figure 2 Code to call the function for a file name

Constants

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

A screen shot of a computer code

Description automatically generated

Figure 3 Example of requesting a file name

Second part – running the interface menu.

A screen shot of a computer program

Description automatically generated

Figure 4 Code to select for a menu option and run the corresponding module

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

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

A screenshot of a computer program

Description automatically generated

Figure 5 Code of possible menu options, const ‘choice\_list’

A screenshot of a computer program

Description automatically generated

Figure 6 Example of running the menu

The similar solution with one dictionary *‘retrieve\_condition’* was applied to filter conditions.

A screen shot of a computer program

Description automatically generated

Figure 7 Code of possible filter conditions, const ‘retrieve\_condition’

First two keys are straightforward: one feature to filter the dataset and several features to display.

The third key is a special flag responsible for choice between searching by one value or bt value range. The initial project requirements do not have such a requirement, but it is helpful for finding devices by 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 user leaves the field blank) and max number to requests.

A computer screen shot of a program code

Description automatically generated

Figure 8 Code of function asking a filename, input\_file\_path()

The example of usage can be found in Figure 3

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

Takes one parameter with all the menu options (‘*action\_list*’) and visualises it in the loop.

A computer screen shot of a program code

Description automatically generated

Figure 9 Code of function visualising menu interface, input\_choice()

The example of usage can be found in Figure 6

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

Both functions take one parameter ‘*key’* and prompt a value (or value range)

A computer screen shot of a program code

Description automatically generated

Figure 10 function input\_value() and function input\_value\_range()

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

Takes two parameters: a path (‘*path’*) and an option what to retrieve (‘*retrieve*\_condition’)

Returns specified fields from the initial dataset filtered by condition.

A screen shot of a computer program

Description automatically generated

Figure 11 Beginning code of function retrieve()

There was a choice how to keep dataset: simple list or dictionary and I opted for dictionary as more flexible. It is easier to maintain when new columns would be added in the dataset.

Theoretically, built-in function *reader* can be replaced by function *DictReader*

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

Simple functions that accept values, field name to filter and the dataset

Returns the dataset filtered by values for the specified key.

A computer screen shot of a program code

Description automatically generated

Figure 12 Code of functions to search by key, find\_by\_key() and find\_by\_key\_range()

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

Takes the filename *‘path’* and specified menu option (‘*choice’*).

Returns the values or charts based on the specified choice.

A screen shot of a computer code

Description automatically generated

Figure 13 Beginning code of function analytics\_visualize()

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

There is a small pre-processing of data: ‘*market\_regions’* is converted to the list datatype for better processing while two additional fields are added (‘*released\_year*’ and ‘*release\_month’*).

Some examples of usage are provided below

|  |  |
| --- | --- |
|  |  |

|  |  |
| --- | --- |
|  |  |

Figure 14 Example of charts

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

Takes dataset, three field names (to filter dataset, to group dataset and to average dataset) and output text.

Prints the list of average values

A computer screen with white text

Description automatically generated

Figure 15 Code of averaging function avg\_group\_two()

Values are converted to lower case to accept the different spelling like ‘Apple’, ‘apple’, ‘APPLE’

A screenshot of a computer

Description automatically generated

Figure 16 Example of analyse of the average price of device

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

Takes dataset, two field names (to group dataset and to average dataset) and output text.

Prints the list of average values

A screen shot of a computer

Description automatically generated

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

Example of usage

A screen shot of a computer program

Description automatically generated

Figure 18 Example of averaging weight for each manufacture

###### 2.2.5.3 Function count\_annualy()

Takes dataset, two field names (to group dataset and to count dataset) and output text.

Prints list of counted values grouped by other feature (like year)

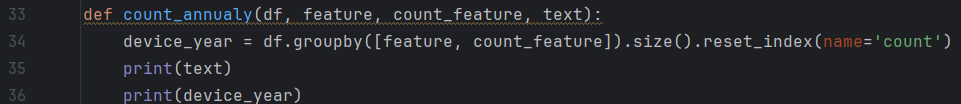


Figure 19 Code of counting function, count\_annualy()

Example of usage

A screenshot of a computer

Description automatically generated

Figure 20 Example of counting device grouped by brand and released year.

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

Takes dataset, two field names (to filter dataset and to count dataset), max number of rows to print and output text. The function also prompts the specified value for field that is filtered.

Prints list of top values. The number of rows defined by max number.

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

A screen shot of a computer program

Description automatically generated

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

Example of usage

A screenshot of a computer

Description automatically generated

Figure 22 Example of top regions for Apple brand

# 3. GitHub Repository Evidence

A screen shot of my private Git repository. The screen shot **clearly shows a history of my commit of project implementation**. There were 23 commits starting 10th November.

I used GitHub to upload data.

A screenshot of a computer

Description automatically generated

Figure 23 Github commits

Sample screen shot of my commit history:

A screenshot of a computer

Description automatically generated

Figure 24 Commit history in GitHub