PYFINANCES

PYFINANCES

PyFinances Requirements Document

Jonathan A. Webb

CONTENTS IN BRIEF

1 Capability Requirements 1

2 Performance Requirements 5

CONTENTS

Introduction ix

1 Capability Requirements 1

1.1 Key Performance Parameters 2
1.1.1 KPP 1. Develop Spending Breakdown 2
1.1.2 KPP 2. Analyze Historical Spending Trends 2
1.1.3 KPP 3. Allocate Checking and Savings Accounts 2
1.1.4 KPP 4. Allocate Paycheck Info 2
1.1.5 KPP 5. Deduct Planned Expenses 3
1.1.6 KPP 6. Deduct Bills 3
1.1.7 KPP 7. Monte Carlo Method 3
1.2 Key Software Attributes 3
1.2.1 KSA 1. Testing 3
1.2.2 KSA 2. Evolvability 3
1.3 Additional Performance Parameters 4
1.3.1 APP 1. Graphical Probability Distribution Functions 4
1.3.2 APP 2. Graphical Cumulative Distribution Functions 4

2 Performance Requirements **5**

2.1 Input Files 5
2.1.1 RunOptions File 5
2.1.2 Daily Expenses File 7
2.1.3 Total Expenses File 7
2.1.4 Deductions File 8
2.1.5 Bills File 8
2.1.6 Planned Expenses File 8
2.1.7 PDF Files 9
2.1.8 CDF Files 9
2.2 Pre-processor 9
2.2.1 Distributions Pre-processor 10
2.2.2 Monte Carlo Pre-processor 10
2.3 Requirements Maping 10

INTRODUCTION

This document serves as the requirements repository for the PyFinances computer code. The PyFinances code will analyze an individuals spending and determine habits and make an estimate for the amount of money in an indivudals checking and savings account as a function of time with estimates for statistical uncertainty. This document will contain the capability, performance and component requirements that guide the development, and use of the software suite.

CHAPTER 1

CAPABILITY REQUIREMENTS

This chapter describes the high-level capability (Level-I) requirements that guide the development and use of the PyFinances software suite. This doducment is divided into Key Performance Parameters, Key Software Atributes, and Additional Performance Parameters. Each term is defined below.

- **Key Performance Parameter (KPP)** - A KPP is an software attribute that is considered critical to the development or use of the software capability. If the final code does not adequately address KPPs then it will not be accepted by the end-user

- **Key Software Attribute (KSA)** - A KSA is a software capability considered crucial in achieving a balanced sokution to KPPs. KSAs generaly cover system maintainability or evolvability and are considered critical to end-user acceptance of the software suite.

- **Additional Performance Parameter (APP)** - APAs are software attributes that are deemed desirable, but not necessary for the development of, or use of a software-suite.

1.1 Key Performance Parameters

The following items are considered necessary capabilities that must be codified in the PyFinances software suite.

1.1.1 KPP 1. Develop Spending Breakdown

The PyFinances software suite *shall* be capable of determining a day by day spending profile for historical spending divided into categories of **gas**, **miscellaneous**, **groceries**, **restaurant**, and **bar**.

1.1.2 KPP 2. Analyze Historical Spending Trends

The PyFinances software suite *shall* be capable of analyzing historical spending trends by the categories of **gas**, **miscellaneous**, **groceries**, `restaurant`, and **bar** spending.

1.1.2.1 KPP 2.1. Develop Probability Distribution Functions The software *shall* develop Probability Distribution Functions (PDF) for each spending category as well as total spending, subtracting the effects of bills, and planned expenses, as well as expected pay deductions.

1.1.2.2 KPP 2.2. Develop Cumulative Distribution Functions The software *shall* develop Cumulative Distribution Functions (CDF) for each spending category as well as total spending, subtracting the effects of bills, and planned expenses, as well as pay deductions.

1.1.3 KPP 3. Allocate Checking and Savings Accounts

The software suite *shall* allocate time dependent values for personal wealth assuming a direct and predictable income.

1.1.3.1 KPP 3.1. Checking Account The software *shall* predict the day by day value of an individual checking account over the user defined time frame.

1.1.3.2 KPP 3.2. Savings Account The software suite *shall* predict the day by day value of an individual savings account over the user defined time frame.

1.1.4 KPP 4. Allocate Paycheck Info

The software suite *shall* determine all pay information necessary to predict the day by day value of checking and savings accounts.

1.1.4.1 KPP 4.1. Determine Pay Dates The software suite *shall* be able to determine the date on which paycheks are released for all reasonable pay allocation schemes.

1.1.4.2 KPP 4.2. Determine Pay Deductions

The software suite *shall* be able to deduction salary from each paycheck to account for medical and dental insurance, life insurance, taxes, 401k, and legal support fees, as well as any other necessary deduction.

1.1.5 KPP 5. Deduct Planned Expenses

The software suite *shall* be capable of deducting planned expenses from the users portfolio on the user defined dates in order to support predictions for the value of checking and savings account.

1.1.6 KPP 6. Deduct Bills

The software suite *shall* be capable of deducting known bills from the users checking and savings account on the desired dates in order to support predictions for the value of each account.

1.1.7 KPP 7. Monte Carlo Method

The software suite *shall* calculate the mean values for checking and savings account as well as the plus and minus 2 standard deviation (2σ) using a Monte Carlo technique.

1.2 Key Software Attributes

The requirements in this section are focused on maintainability and evolvability.

1.2.1 KSA 1. Testing

The software *shall* be built with unit tests and regression testing for all primary functions, classes, and components. Unit and regression testing *shall* be used to validate performance requirements, which will validate all capability requirements that map to those performance requirements.

1.2.2 KSA 2. Evolvability

The software *shall* be developed with object oriented and component oriented practices for the purpose of ensuring that the software can be evolved in the future to account for multiple bank accounts.

1.3 Additional Performance Parameters

The following attributes can increase the information that the user can extract for their spending portfolios. These attributes are considered advantageous but not necessary.

1.3.1 APP 1. Graphical Probability Distribution Functions

The software suite *should* allow a user to view their spending PDFs as .png documents

1.3.2 APP 2. Graphical Cumulative Distribution Functions

The software suire *should* allow a user to view thei spending PDFs

CHAPTER 2

PERFORMANCE REQUIREMENTS

This chapter describes the level-II requirements that provide detail to how the software suite will meet the intent of the level-I requirements. The verification of these level-II requirements will validate all level-I requirements that they map to.

2.1 Input Files

The following input files are required for the operation of the PyFinances software suite.

2.1.1 RunOptions File

The program *shall* use a file titled `RunOptions.txt` that guides the exection of the software suite. This file contains all information necessary to guide the execution of the PyFinances computer program. The file *shall* contain the following user inputted variables;

1. **Run Monte Carlo:** True or False

2. **Sample Size:** Describes the number of Monte Carlo iterations used to determine a stochastic estimate of account values

3. **Start Date:** Describes the date where financial estimates should beggin.

4. **End Date:** Describes the date after which financial estimates should cease.

5. **Checking Start Value:** The initial value of the checking account on the **Start Date**.

6. **Savings Start Value:** The initial value of the savings account on the **Start Date**.

7. **Annual Salary:** The expected annual salary, assuming a constant pay stream.

8. **Pay Frequency:** The frequency that pay allocations are made. The options *shall* be 'weekly', 'two weeks', 'bi-monthly', and 'monthly'.

9. **First Pay Date:** The first date when a pay allocation is made. This date must be align with the 15th and the last day of the month if 'bi-monthly' is chosen, and the end of the month if 'monthly' is chosen.

10. **Run Histogram:** True or False

11. **Bins:** The number of bins used in the probability and cumulative distribution functions.

12. **Hist Start Date:** The date within the historical spending trend that should be used as the start point for the development of a histoggram.

13. **Hist End Date:** The date within the historical spending trend that should be used as the end point for the development of a histogram.

14. **Daily Expense File:** The name of the expense history file that will be used to generate a daily spending file to include the path-length to the file.

15. **Deductions File:** The name of the file .csv file containing pay deduction information.

16. **Expenses File:** The name of the .csv file containing expected expenses and account additions to include the path length.

17. **Bills File:** The name of the .csv file containing expected bill information, to include the path length.

If **Run Monte Carlo** is False, then **Run Histogram** must be True. In this case, the **bins**, **Hist Start Date**, **Hist End Date** and **Daily Expense File** attributes must be populated. If **Run monte Carlo** is True, then all fields must be populated.

2.1.2 Daily Expenses File

The PyFinances software suite *shall* be capable of reading a file titled `Daily_Expenses.csv`. The `Daily_Expenses.csv` file contains a list of all expenses used to create a statistical spending profiles. The `Daily_Expenses.csv` file has the following column headers.

1. **Date** - The date of the transaction in the MM/DD/YY format
2. **Checking_Debit** - The amount subtracted from the checking account.
3. **Checking_Addition** - The amount added from the checking account.
4. **Savings_Debit** - The ammount subtracted from the savings account.
5. **Savings_Addtion** - The ammount added to the savings account.
6. **Expense_Type** - The type of expense, which can be one of the following entries, Bills, Misc, Groceries, Gas, Bar, Restaurant, Paycheck, Fed Taxes, State Taxes, Planned Expense.
7. **Vendor** - The vendor from which the item was purchased or a credit was given.
8. **Description** - A description of the purchase or credit.

i
The `Daily_Expenses.csv` file can contain multiple entries for the same day.

2.1.3 Total Expenses File

The PyFinances software *shall* be capable of reading and writing a file titled the `Total_Expenses.csv` file. The `Total_Expenses.csv` file contains a break-down for the total amount of money spend on each day for the following categories, which also act as colum headers.

1. **Date** - The expense or credit date in the MM/DD/YY format
2. **Misc** - Miscellaneous expenses not covered in other topics, excluding taxes
3. **Bills** - Money spend on bills
4. **Groceries** - Montey spent on groceries
5. **Gas** - Money spent on automative gasoline
6. **Bar** - Any money spent specifically on alcohol
7. **Restaurant** - Any money spent eating at restaurants
8. **Planned Expense** - Any money spent on a topic that was planned. Christmas gifts are a good example.

2.1.4 Deductions File

The PyFinances program *shall* read a deductions file that contains all deductions from a paycheck, which can include federal and state taxes, 401k, medical benefits and other information. This file can be named whatever the user wishes it to be, but it must be defined in the `RunOptions.txt` file under the name **Deductions File** and it must be a .csv file. This file will contain the following headers. The assumption made for these requirements is that the total paycheck allocation will be allocated to the checking account, and any amount that should be placed in the savings account will be deducted as a bill. In addition, it is assumed that the money is deducted on the dates of pay allocation.

1. **Deduction** - The financial amount to be deducted from the paycheck
2. **Description** - A description of the deduction

2.1.5 Bills File

The PyFinances program *shall* read a bills file, which contains all annual, usually monthly bills to be deduction from a checking and savings account. This file can be named whatever the user wishes it to be, but it must be defined in the `RunOptions.txt` file under the name **Bills File** and it must be a .csv file. This file will contain the following headers.

1. **Date** - The Date that bills will be deducted in the format MM/DD/YYY.
2. **Checking_Debit** - The amount to be deducted from the checking account.
3. **Checking_Addition** - The amount to be added to the checking account.
4. **Savings_Debit** - The amount to be deducted from the savings account
5. **Savings_Addition** - The amount to be added to the savings account
6. **Description** - A description of the transaction

2.1.6 Planned Expenses File

The PyFinances program *shall* read a Planned Expenses file, which contains all planned expenses that are not a paycheck deduction or a bill. This file can be titled whatever the user wishes, but it must be defined in the `RunOptions` file under the name **Expenses File** and it must be a .csv file with the following headers.

1. **Date** - The Date that bills will be deducted in the format MM/DD/YYY.
2. **Checking_Debit** - The amount to be deducted from the checking account.
3. **Checking_Addition** - The amount to be added to the checking account.

4. **Savings_Debit** - The amount to be deducted from the savings account
5. **Savings_Addition** - The amount to be added to the savings account
6. **Description** - A description of the transaction

2.1.7 PDF Files

The PyFinances program *shall* be capable of creating and reading a file titled `pdf_data.csv`, which contains the values of each probability distribution bin for each expense type. The file should have a number of rows equal to the number of bins and the following header structure;

1. **bins** - The bin number
2. **groceries** - The bin value for groceries
3. **misc** - The bin value for miscellaneous
4. **bar** - The bin value for bar
5. **restaurant** - The bin value for restaurants
6. **gas** - The bin value for gas

2.1.8 CDF Files

The PyFinances program *shall* be capable of creating and reading a file titled `cdf_data.csv`, which contains the values of each cumulative distribution bin for each expense type. The file should have a number of rows equal to the number of bins and the following header structure;

1. **bins** - The bin number
2. **groceries** - The bin value for groceries
3. **misc** - The bin value for miscellaneous
4. **bar** - The bin value for bar
5. **restaurant** - The bin value for restaurants
6. **gas** - The bin value for gas

2.2 Pre-processor

s section describes the requirments for the pre-processor and its subsequent components

2.2.1 Distributions Pre-processor

The Distributions pre-processor shall read in the users historical spending data and transform it into .csv files containing PDF and CDF information. This pre-processor *shall* be invoked if the **Run Histogram:** option is listed as True in the `RunOptions.txt` file.

2.2.1.1 Read Daily_Expenses.csv The distributions pre-processor *shall* read the `Daily_Expenses` file described in Section 2.1.2. This pre-processor.

2.2.1.2 Create Total_Expenses.csv The distributions pre-processor *shall* transform the information within the `Daily_Expenses.csv` file into the `Total_Expenses.csv` file described in Section 2.1.3. This file should contain a day by day account of how much money is spent in total for each of the categories to include `bar`, `groceries`, `misc`, `gas`, and `restaurant` as well as `planned`.

2.2.1.3 Create pdf file The distributions pre-processor *shall* read the data in the `Total_Expenses.csv` file and transform it into the pdf file described in Section 2.1.7.

2.2.1.4 Create cdf file The distributions pre-processor *shall* read the data in the `cdf_data.csv` file and transform it into the cdf file described in Section 2.1.8.

2.2.1.5 Plot Data The distributions pre-processor *should* produce a set of pdf and cdf plots based on the data in the `pdf_data.csv` and the `cdf_data.csv` file.

2.2.2 Monte Carlo Pre-processor

The Monte Carlo pre-processor *shall* prepare all data necessary to run the Monte Carlo simluation, not already produced in Section 2.2.1.

2.2.2.1 Validate Distributions The Monte Carlo pre-processor *shall* verify that the `cdf_data.csv` file exists. If the file does not exist, then the program *shall* envoke the Distributions Pre-processor to create the correct files.

2.2.2.2 Create Date List The Monte Carlo pre-processor *shall* produce a date list covering every day from the **Start Date:** and **End Date:** described in the `RunOptions.txt` file. The dates within the list *shall* be in the format MM/DD/YYYY.

2.2.2.3 Create Pay Date List The Monte Carlo pre-procssor *shall* produce a list of every pay date, where each date *shall* be in the format MM/DD/YYYY.

2.2.2.4 Create Input Containers The Monte Carlo pre-processor *shall* produce a, or multiple containers to hold the information read from the `bills.csv`, `deductions.csv`, and `planned_expenses.csv` files.

2.3 Requirements Maping

Table 2.1 shows the mapping of level-I to level-II requirements.

Table 2.1 Level-I to Level-II requirement mapping.

Level-I Requirement	Level-II Requirements
KPP 1. Develop Spending Breakdown	2.1.1. RunOptions file 2.1.2. Daily Expense File 2.1.3. Total Expenses File 2.2.1. Distributions Pre-processor 2.2.1.1. Read Daily_Expenses.csv 2.2.1.2. Create Total Expenses.csv 2.2.1.3. Create pdf file 2.2.1.4. Create cdf file
KPP 2. Analyze Historical Spending Trends	2.1.1. RunOptions file 2.1.2. Daily Expense File 2.1.3. Total Expenses File 2.2.1.3. Create pdf file 2.2.1.4. Create cdf file
KPP 2.1. Develop Probability Distribution Functions	2.1.1. RunOptions file 2.1.7. PDF Files 2.2.1.3. Create pdf file
KPP 2.2. Develop Cumulative Distribution Functions	2.1.1. RunOptions file 2.1.7. CDF Files 2.2.1.4. Create cdf file
KPP 3. Allocate Checking and Savings Accounts	2.1.1. RunOptions file 2.2.2.4. Create Input Containers
KPP 3.1. Checking Account	2.1.1. RunOptions file 2.1.5. Bills File 2.1.6. Planned Expense File 2.2.4. Create Input Containers
KPP 3.2. Savings Account	2.1.1. RunOptions file 2.1.5. Bills File 2.1.6. Planned Expense File 2.2.4. Create Input Containers
KPP 4. Allocate Paycheck Info	2.1.1. RunOptions file 2.2.2.3. Create Pay Date List 2.2.2.4. Create Input Containers
KPP 4.1. Determine Pay Dates	2.1.1. RunOptions file 2.2.2.3. Create Pay Date List
KPP 4.2 Determine Pay Deductions	2.1.1. RunOptions file 2.1.4. Deductions File 2.2.2.4. Create Input Containers
KPP 5. Deduct Planned Expenses	2.1.1. RunOptions file 2.1.6. Planned Expenses File 2.2.2.4. Create Input Containers
KPP 6. Deduct Bills	2.1.1. RunOptions file 2.1.5. Bills File 2.2.2.4. Create Input Containers
KPP 7. Monte Carlo Method	2.1.1. RunOptions file 2.1.7. PDF Files