Andrew Feather - af2849
Mat Federer - mrf2157
PLT W4115
7/11/2016

Project Proposal

I. Language Description:

We plan to implement a scripting language language focused on financial calculations and economic projections to make the jobs of financial professionals and economic researchers a bit easier.

The basic language will focus on making it easy to read in spreadsheets, manipulate columns and rows of data, perform common calculations, plot the data on a chart, and write back to a new spreadsheet. The hope is that the language can be used by financial and economic-research focused shops to easily construct customized programs to input and output the information they need from their most common data input sources. They will also be able to quickly generate any information they need with built-in functions for common calculations and the customizable 'function' object. Shops can use these capabilities to build their own, specific calculations.

Our language will yield agile capabilities for financial professionals and economic researchers to build custom programs that can scrape data into a short program, run the calculations they need and output whatever charts, spreadsheets or other documents they would like to return.

Array creation, use and concatenation will also be made intuitive and easy, much in the same way it works in Matlab. The compiler will easily handle the creation of 2d arrays simply by the user giving '[][]' at the end of the instantiated array's name. However, the goal of the standard library will be to eliminate the need to create arrays manually as much as possible by making simple array operations, such as reading in columns of data easy via basic function calls.

II. Purpose:

The language will quickly build programs to run customized financial and economics calculations from common financial documents.

A standard library will be included with built-in objects (spreadsheet, account, company, earnings_report) that allow users to easily read in spreadsheets with a list of defaults and prompts that make it very easy to read in and write values to and from spreadsheets and use common operations on objects such as accounts, companies and earnings reports.

Economics calculations will be supported by an economics library. This library will redefine functions for basic calls like demand and price. Larger formulas like Income Elasticity or Cross-Price Elasticity of Demand must be user defined. Our goal is that by defining functions for

only basic economics variables, our language will accelerate fiscal calculations while avoiding the Standard Library Syndrome.

The most common programs will involve reading in information from common types of earnings reports and outputting formatted information with user-specified calculations. The language and its libraries can also be utilized to append information to an existing report. Ideally, the language will easily be used to manipulate a company's internal database of hundreds or thousands of reports. Our intention is that such natural interaction will help expedite decision making and assure the preservation of long term data.

III. Parts and Function:

Integers/Floating Point Integers: Same notation as C
Strings: Denoted by singles quotes or double-quotes.

- You can offset quotes within quotations by using /"\. When using double quotes, single quotes will print with no need for any extra characters.
- Printing objects works similar to python "This is the name of an object: {}".objects_to_print(this)

Arrays: Our language will allow for the easy manipulation of arrays, much in the same way Matlab implements its array structures. Both one dimensional and two dimensional arrays will be available, accessible via 'name[]' and 'name[][]' respectively.
Comments: Same as C - single-line: "//", multi-line: "/* */"
Objects: Objects require the keyword obj and require at least two attributes. These are similar to classes in Java.
Functions: Functions use the keyword "func" and require the user to indent, as with the rest of the code.

IV. Sample Source Code:

A. Using Standard Library

```
Import standard_library

Func days_sales_in_inventory(inventory_turnover) =
        Return 365/inventory_turnover

data = File(bank_spreadsheet.xls)

label_column = get_label_column() //requests label column from user

temp_company = (name="Pfizer")

company_earnings_report = earnings_report(label_column, data)
```

```
//pulls data based on the label column input by the user above

temp_Assets_label = ‘Current assets:’
temp_Liabilities_label = ‘Current liabilities:’
temp_Equity_label = “Shareholders’ equity:”
//double quotes allow you to ignore apostrophes

temp_company.add_earnings_report(id=1, month=”January”,company_earnings_report,
assets_label=temp_Assets_label, liablities_label=temp_Liabilities_label,
equity_label=temp_Equity_label)

print(“Days’ sales in Inventory
{}”.objects_to_print(days_sales_in_inventory(temp_company.inventory_turnover))

New_spreadsheet = spreadsheet(“new_spreadsheet.xls”) // creates a new spreadsheet

New_spreadsheet.write_common_earnings_report_values(temp_company.earnings_rep
ort(1))
/*this writes all common earnings report calculations to an output spreadsheet using the
earnings report with id=1*/
```

B. Using Economics Library

```
/* Decision making program printing the Quarter 1 to Quarter 2 own-price elasticity
calculation to the user */

Import standard_library
Import econ_library as econ

data1 = File(bank_spreadsheet.xls)

temp_company = (name=”Pfizer”)

prevquarter_demandinfo = get_label_column() // requests label column from user for last
quarter column with demand info
prevquarter_priceinfo = get_label_column() // price info

thisquarter_demandinfo = get_label_column() // requests label column from user for
current quarter column with demand info
thisquarter_priceinfo = get_label_column() // price info

prevquarter_reportD = earnings_report(prevquarter_demandinfo, data)
 //pulls data for previous quarter demand
```

```
prevquarter_reportP = earnings_report(lastquarter_priceinfo, data)
//previous quarter price

thisquarter_reportP = earnings_report(thisquarter_demandinfo, data)
//pulls data for current quarter demand

thisquarter_reportD = earnings_report(thisquarter_priceinfo, data)
//current quarter price

prev_demand = econ.demand(prevquarter_demandinfo)
// econ.demand calls the demand function from the economics library

prev_price = econ.price(prevquarter_priceinfo)
// econ.price call price function from the economics library
this_demand = econ.demand(thisquarter_demandinfo)
this_price = econ.price(thisquarter_priceinfo)

Func own_price_elasticity(prev_demand, prev_price, this_demand, this_price) =
return (((this_demand - prev_demand) / (this_price - prev_price)) * (this_price /
this_demand))

print("Quarter 1 to Quarter 2 Own-price Elasticity: {}
".objects_to_print(own_price_elasticity(prev_demand, prev_price, this_demand,
this_price))
```