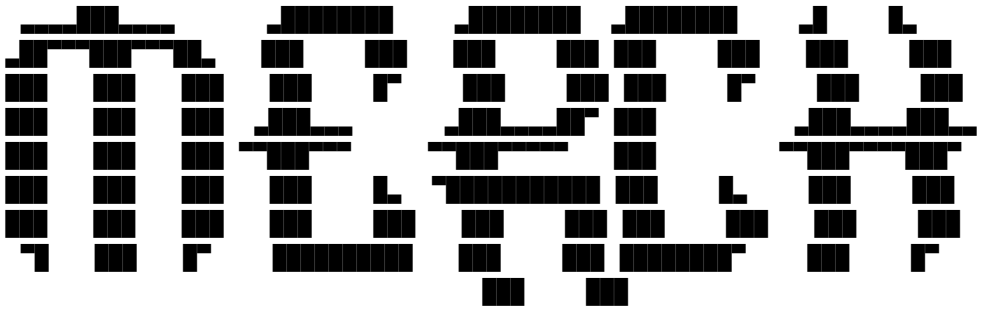

White Paper

Compilers - CSC 492 - Fall 2018

Dr. Paul Hinker

Ryan McCaskell, Lyndon Engel, Jeff Ross,

Joe Ceritelli, and Jimmy Hinker

Domain of application

What is the problem?

The problem we are addressing is in direct correlation with time expended each day on repetitive tasks that could be done through programming. The problem then presented would be the lack of knowledge or time committed to learn the ins and outs of programing. Specifically, we look to focus on the tasks of searching for information on the stock market that directly relates to an individual's finances.

How do we plan to fix it with MERCH(50,000ft)

The MERCH programming language is designed to meet the challenges of a user that is adept at confronting the behemoth of the stock market on a monthly to daily basis. The biggest hurdle to tackle is making MERCH feel familiar to someone who is not already knowledgeable on how to write a “Hello World” program in any language. The MERCH programming language was born from counting up the wasted hours of searching the stock market for specific numbers. The goal was to make a language that can write scripts to be simply edited and expanded to find the numbers you want on a daily basis, without the repetitive time consumption of everyday stock market analysis.

Beginnings of MERCH

The MERCH domain specific language originated as a college project for a compilers course. The MERCH project is advised by Dr. Paul Hinker and designed/built by students Ryan McCaskell, Lyndon Engel, Jeff Ross, Joe Ceritelli, and Jimmy Hinker.

Main Feature(s) of the MERCH Language

The main features of this language include finding extensive stock information from short concise sentences, and allowing users who are adept at using stock data collection systems to create a simple to use and reusable system for collecting stock data.

Design goals of MERCH

MERCH is Understandable

The idea of MERCH is to create a simple, understandable language which can be used by those with zero background in programming, but have an understanding of the stock market and the associated terminology. Virtually anyone with basic knowledge in the stock market should be able to pick up the language and start performing analytics on the stock market.

MERCH is Portable - anywhere python 3 is

In order to run, the DSL will require Python to be installed on the computer. Upon installation of the DSL, if python is not available, it will be downloaded and installed alongside the DSL, making it portable.

Programming Paradigm

MERCH is a scripting language. The intent is for simple reusable code that someone can use daily to help expedite tedious or repetitive tasks common while performing stock analysis.

Program Structure

The goal for MERCH is that a line of code will have a clean natural sounding feel when read aloud. These sentences will be passed into a text file so that they can be parsed to find tokens in the sentence, which are then passed into a variable stock array to form Lexemes.

Memory management model

The memory for MERCH will be handled internally. Users of this language would not be expected to understand how memory is managed or handled. The underlying interpreter will be using data structures and memory management of Python3 within its implementation.

Data

Stock information is gathered from the NASDAQ website during runtime. The user is not able to enter information during runtime. All data is expected to be provided before runtime, provided via variables or directly in statements.

Data types

The data types that would be necessary for this project would include a stock type that people could define in an array with a syntax similar to “My stocks are AAPL, GOOGL”. Additionally, the project would require support of some kind of cutoff value that could be used to help the user make educated decisions about purchasing and trends. One example would be saying that if the return from percent change was below 4%, write out to the screen “do not invest”.

Character set and lexemes

The character set is made up of both upper and lower case alphabetic characters [A-Z, a-z]. The numeric characters [0-9]. As well as the special characters [+,-,*,/,., <, >, =, %, comma, space].

Operators

Future goals for the language will include boolean operators to allow for comparison. Boolean operations would allow the user to create MERCH scripts that output results from the comparison between two sentences in our language.

Identifiers and Variables

The language will allow for the creation of variables using keywords to save data within the variable. There will be predefined methods that allow actions to be performed on variables and stock names. Variables names can only be defined with a combination of upper and lowercase letters and numbers.

Expressions

Mystocks are AAPL, GOOGL

Mystocks is defined as a token that saves other tokens inside of it for ease of use in future locations. AAPL and GOOGL are example tokens that define stock names.

Write string
Write Hello World

Print statements will consist of a simple write command followed the the information to be output to the screen.

Get method <days> for stocks

Method can be one of the following:

- <days> moving average - moving average cannot be followed by <days>
- Percent change
- High price
- Low price
- Opening price
- Closing price
- Price
- Volume

Days is a whole integer number, followed by the word day or days
This parameter is optional and will default to 1 day for all of the methods except for moving average which will default to 30 days
Ex. 30 days

Stocks is a variable name or a list of stock ticker symbols

Wait <time>
Wait 1 hour

The wait functionality will provide delays in program execution to allow users to create programs that can be repeated without the need for re-executing the program.

Flow of control

MERCH is a language linear top to bottom program. Conditionals and looping will allow for automatic time interval repetition or conditionals based on behaviors of received stock information. Furthermore, conditionals will allow for use of boolean operations within an If Then statement.

Functions and process abstractions

There will be no functions or abstraction at this point in time. A Program will be written in a text document. Programs can be compiled together so that all variables may be used across all programs.

Objects and data abstractions

As of the beginning we have not added additional abstraction beyond base data types, however during future development we are open to implementing them and keeping it in mind in current development.

Parallelism / Concurrency

Initial designs for MERCH will not include concurrency. However, the language will be built with concurrency in mind so that later releases could easily support this feature. The current scope and complexity of the language should not require parallelism to be effective or efficient. In future releases, if implemented, parallelism is not a feature that the user would be expected to be knowledgeable of, and would be handled by the language itself.

Event Handling

MERCH does not have any required events or event handling. However in the case of events needing to be handled python2.7 will pick up that cross and handle those problems.

Exception handling

The user will not need to catch or handle exceptions on their own. Due to the primary user not being expected to possess programming experience, MERCH will attempt to interpret input from the user and execute until completion or error. If an error is found, MERCH will report the problem to the user, that they can fix any grammar or syntax issues. When most errors occurred, execution will not cease, unless the error is critical to execution.

Because MERCH is a translator using python2.7 exception handling is handled by python2.7.

Further Reading

Python 2.7 - https://www.python.org/doc/
Project Outline - https://www.mcs.sdsmt.edu/csc492/Project/