University of Puerto Rico
Mayaguez Campus
Electrical and Computer Engineering Department
ICOM 4036 (Programming Languages)
Prof. Wilson Rivera

Angélica Santiago: angelica.santiago4@upr.edu
Luis Santiago: luis.santiago56@upr.edu
Nelson Martinez: nelson.martinez3@upr.edu
Elnois Norat elnois.norat@upr.edu

Phase 3: Évents Final Report

July / 02 / 2017

INTRODUCTION

Évents is a scripting language that allows you to schedule or manage your events in just one line. It was motivated by other similar applications with the purpose to make the process more easy by using simple commands and parameters, very similar to a human language.

LANGUAGE TUTORIAL

Évents provides the basic features needed at the time of scheduling your events as add, remove and edit. It allows you to see a particular event, events on a specific day or time, and all events you have in the agenda. It is important to know the status of your events, here you can mark your event or attach some comments to let you know its status.

First run Events.py. To start enter a username.

Then enter any command available. For the first time, you will need to use the command 'add'. For example, these are the three options available:

```
add evento 01/01/17 01:35
```

```
add fiesta 03/07/17
```

or just

```
add viaje
```

To edit “evento” time insert:

```
edit evento 12:12
```

Now, if you want to delete an event just enter:

```
remove fiesta
```

or if you want to mark as done,

```
mark evento Done
```

Finally, you can use the ‘view’ command in eight different ways, to view “evento” event,

```
view evento
```

the output will be

```
---------01/01/17-----------

evento    (12:12)   - Done
```

or by entering ‘view’ without parameters the display will be all your events.

LANGUAGE REFERENCE MANUAL

Command	Description	Command and parameters order
add	Add an event at any valid calendar date	**add** [event name] [date] [time] **add** [event name] [date] **add** [event name]
remove	Deletion of any event on a given valid calendar date	**remove** [event name]
edit	To adapt events as needed in case of changes in schedule	**edit** [event name] [date] [time] **edit** [event name] [date] **edit** [event name] [time]

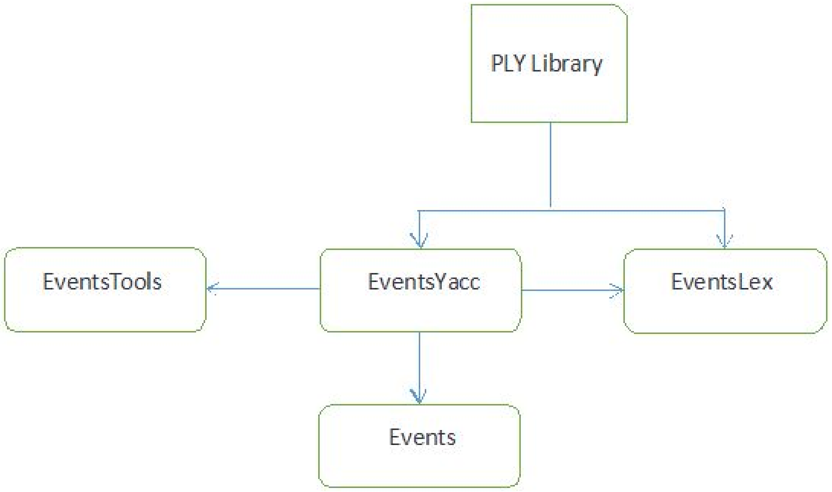
mark	Allow any event to be marked as completed or other status states.	**mark** [event name] [status]
view	Allow all events in a valid calendar date to be view	**view** [event name] [date] [time] **view** [event name] [date] **view** [event name] [time] **view** [date] [time] **view** [event name] **view** [date] **view** [time] **view**

Date format: dd/mm/yy

Time format: hh:mm

LANGUAGE DEVELOPMENT

Translator Architecture



Module Interaction

Initiation occurs at Events.py. This module allows for the continuous input of commands by the user needed for the management of events. This input state will continue until the quit command is input. The inputs are then passed to the parser imported from EventsYacc.py. The same EventsYacc.py will be importing Yacc(Yet Another Compiler-Compiler) from the PLY(Python Lex Yacc) library. The functions of the Yacc parser are the ones responsible for defining the correct grammar of the inputs entered by the user.

The token specifications are found defined in EventsLex.py and consist of the name of the token and the regular expression needed for the inputs. The Yacc parser is responsible for verifying all input that all inputs match the specifications of the tokens. The parser then sends those verified inputs into EventsTools.py where it is used in the appropriate manner depending on the action desired, be it the adding, editing, marking, deleting or viewing of events.

All changes are saved onto pickle files created in EventTools.py under the name of the user.

Software Development Environment

PyCharm and Visual Studio Code- Évents main development IDEs.
Google Drive - Main communication method between teammates. Secure Cloud storage and file backup.
Github - A web-based Git repository hosting service.

Test Methodology

All testing was done on EventsTest.py. Every function of EventTools.py was placed into EventsTest.py and activated to make sure every function performed as necessary. Results were verified to run both of the IDE console and on the terminal.

Conclusion

The development of Évents was not a simple endeavor. The one of biggest roadblocks we had could have been making sure every possible option for the input and editing of commands for the language. More specifically all the possible options to use add, mark, edit and view. But and even bigger roadblock was simply getting everything to compile correctly and learning the proper use of the PLY library for the parser. However, it cannot be denied that this project greatly helped improve our knowledge of the python programming language and the challenge that goes into the construction of a parser and lexical along with the scope of complexity that goes into even the most simple of programming languages. These are lessons that will serve us well into the future in our careers as our learning experiences here transfer into our professional lives.