FLOPY V0.1 DOCUMENTATION

Zovin Khanmohammed

Texas Tech University
CS1411.001
Fall 2014

Introduction

Welcome to my beautiful Programming Principles 1 semester project called FloPy (pronounced Flo-Py)! This program is designed to visualize code for the new Python programmer. This is a proof-of-concept and it was programmed entirely in Python using IronPython, an open-source implementation of Python 2.x with .NET integration. .NET was primarily used for the GUI, using the Windows Presentation Foundation (WPF). Because of the lack of time to develop this, there are a few kinks in the graphics and programming that need to be worked out. Those kinks DO NOT affect overall functionality, except when opening files, which I will address later. Since this is considered a pre-release version (0.1), there are a few features missing like charting loops and elifs.

Interface Overview

FloPy consists of three main columns:

- The Toolbar
- The Flow Chart
- The Editor

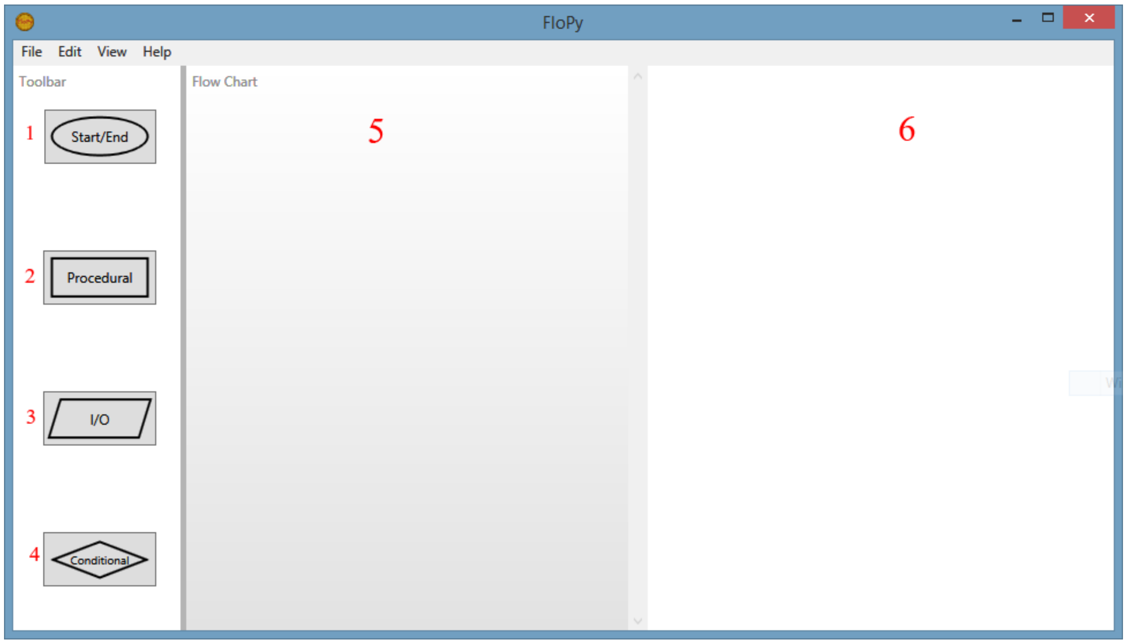


Figure 1

The Toolbar consist of four flow chart components (Figure 1):

1. Create Ellipse – This button will create either the start or end of a function.
2. Create Rectangle – This button will create procedural statements.
3. Create Parallelogram – This button will create procedural input/output statements like print(), input(), or open()
4. Create Diamond – This button will create a conditional statement, this doesn't include loops or elifs, at least in v0.1.

The Flow Chart, indicated by 5 in Figure 1, will display the flow chart for the python text created in the Editor. It will automatically update, but a manual update can be initiated by:

- Selecting the View Menu
- Select 'Refresh Flow Chart'

The Editor, indicated by 6 in Figure 1, will display your python code. You can edit code here manually, or code will be generated by creating the Flow Chart components. The flow chart will automatically update every time you leave the Editor. **THE EDITOR REQUIRES TABS FOR WHITESPACE INDENTS.**

Flow Chart Overview

Figure 2

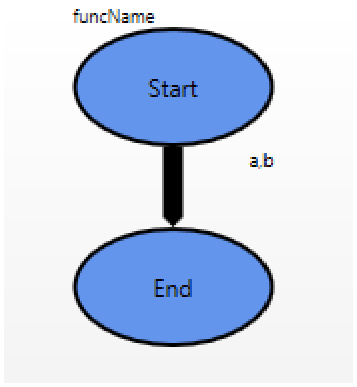


Figure 2 is the ellipse flow chart component. This has two different versions: the start ellipse and end ellipse. The start ellipse displays the function name (represented by "funcName" in Figure 2) and the function variable parameters (represented by "a,b" in Figure 2). The end ellipse designates the end of the defined function.

Figure 3

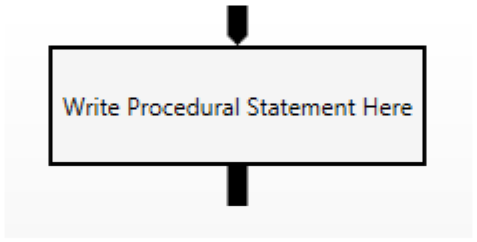


Figure 3 is the rectangle flow chart component. This component designates procedural statement, such as "a = 1".

Figure 4

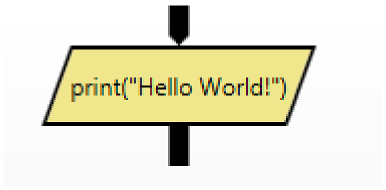


Figure 4 is a parallelogram flow chart component. This component designates that the procedural statement is an input/output statement, such print(), input(), or open(). Due to time constraints and a few graphical issues, the parallelogram is replaced with a regular procedural rectangle when refreshed

Figure 5

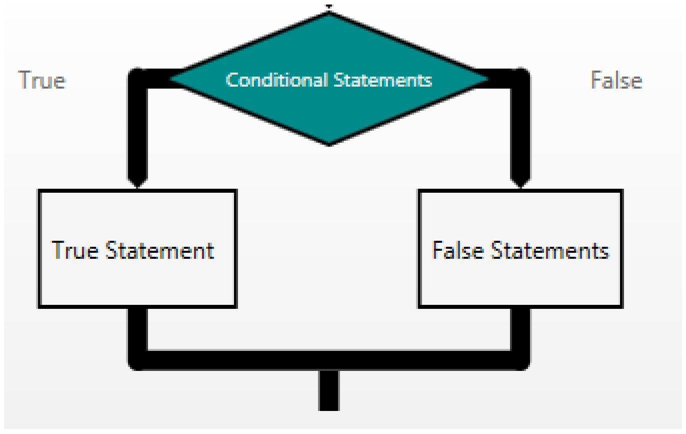
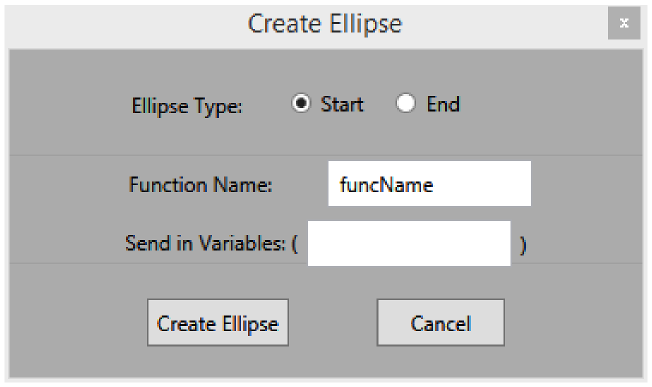


Figure 5 is a diamond flow chart component with corresponding true/false statements. This component has a few issues after refreshing with long if-else statements. Due to time and graphical issues, I was not able to fix this kink in v0.1.

Create Ellipse Prompt

Figure 6



When the Create Ellipse button is selected, the Create Ellipse Prompt (Figure 6) will appear. Here you can select whether this is a start or end ellipse.

If you select START (default), you can then name the function in the "Function Name" Textbox. You can also give the function parameters in the text box surround by the function parenthesis. Parameters will be placed in the code the exact way they are typed here.

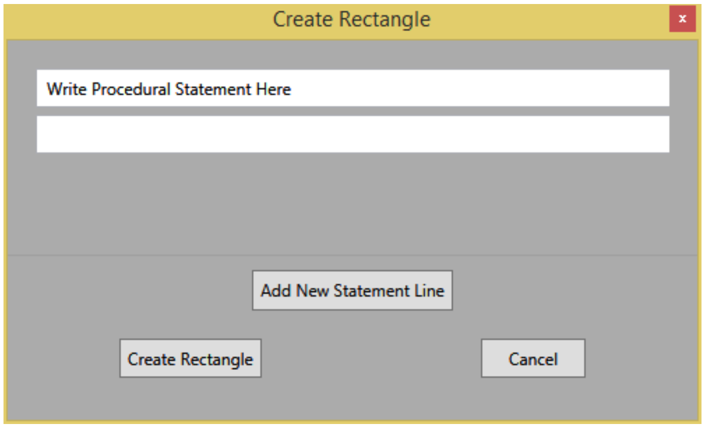
If you select END, the two textboxes will disable.

Once you have selected the information for your ellipse, click "Create Ellipse" to place the code and new ellipse in the main window.

NOTE: You cannot edit the ellipse in the flow chart viewer, but adjustments can be made in the editor.

Create Rectangle Prompt

Figure 7



When the Create Rectangle button is pressed, the Create Rectangle Prompt (Figure 7) will appear. This prompt allows you to type as many procedural statements as you want it the rectangle. Tabs are not allowed here due to programming restrictions and other repressed bugs.

To create more statements, click the "Add New Statement Line" button. A new blank line will appear underneath the last. All blank lines will be ignored.

Once you are satisfied with your statements, you can create the rectangle by pressing "Create Rectangle".

NOTE: You cannot edit the rectangle in the flow chart viewer, but adjustments can be made in the editor.

Create Parallelogram Prompt

Figure 8

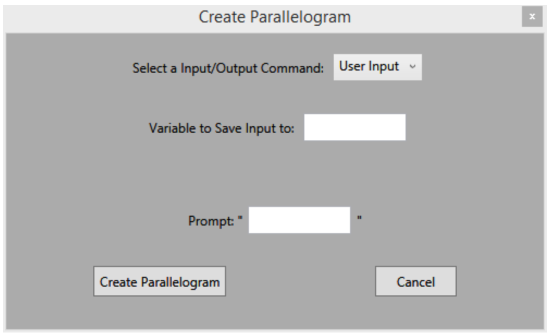


Figure 8 shows the first option of the Create Parallelogram Prompt. This is the default input() operation.

First give the variable, which you want the input to be saved to, and then you can give the prompt, which is already surrounded by quotes. This may change in future versions, but the string itself is only required.

Click the "Create Parallelogram" button to confirm your choice.

Figure 9

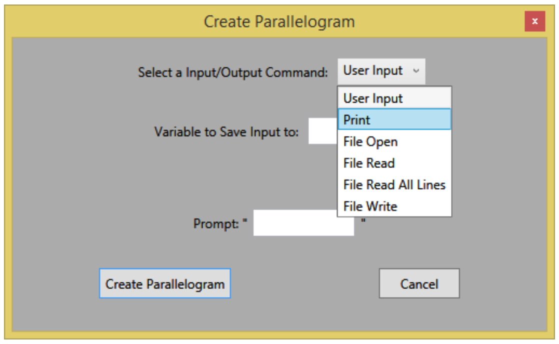


Figure 9 highlights the different input/output operations currently supported by FloPy's parallelogram. The interface will update accordingly. You can only choose one for each Parallelogram.

Figure 10

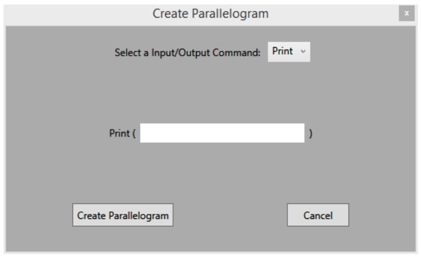


Figure 10 is a simple print() operation. Notice the parenthesis that allows you to put whatever you want printed in the console.

Figure 11

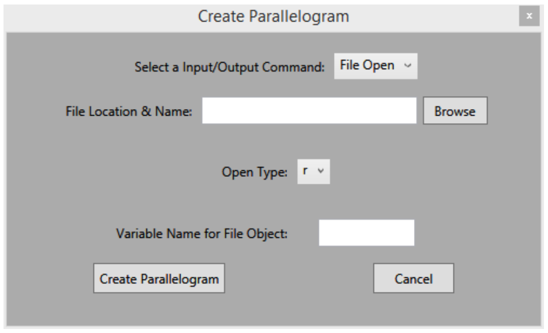


Figure 11 shows the File Open operation. This view allows you to enter the file name and location (relative or full) or browse for a file. Browse will return the full address of the selected file.

After selecting the file, choose how you want python to open the file: "r", "r+", "w", etc.

After the selecting the open type, give the variable name you would like to save the file object to.

Figure 12

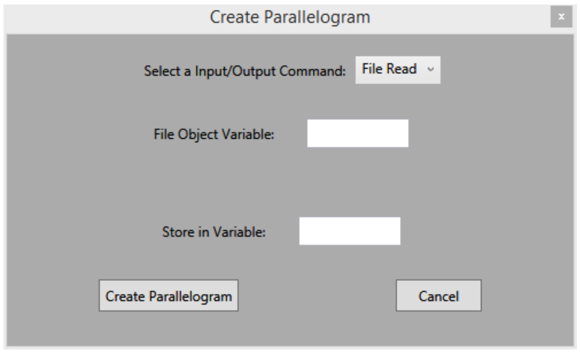
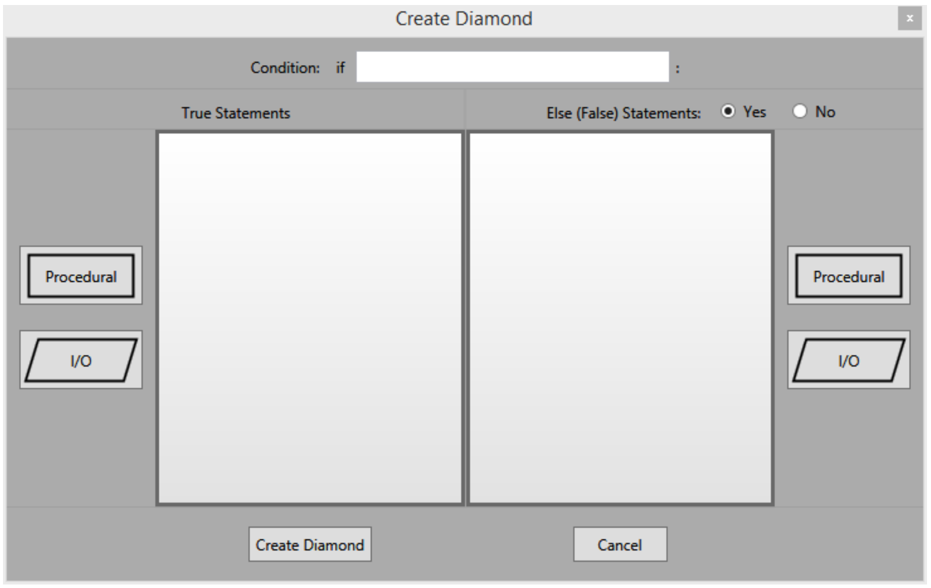


Figure 12 gives the similar layout for the remaining 3 options. All you need is to give the file object's variable name and the variable you would like to store the line in.

Note: The parallelogram is recreated when the flow chart is refreshed manually nor after editing code in the Editor. It is replaced with a regular rectangle currently, but all the code should still be there.

Create Diamond Prompt

Figure 13



Finally, the create diamond prompt will appear when the create diamond option is selected.

First, the condition should be entered in the very top textbox surrounded by the "if" and ":" like a regular python conditional statement is made.

Next, you can create the procedural statements for when the condition is true on the LEFT column. You can click on either the "Procedural" or "I/O" (input/output) buttons (like the main window with prompts) on the left to create the relative component in the True or left column. Only procedural statements are supported for conditionals at the moment.

If you want an else statements for the conditional, select the "Yes" radio button to enable the RIGHT column. Creating the procedural statements is the same has the left, but use the buttons on the right hand side.

Click "Create Diamond" to place the component into the main flow chart.

Note: The diamond has already taken too much system resources, and still has many Graphical kinks, such as the bottom and top lines not completely connecting. To fix this requires multi-threading programming, which has not been implemented into FloPy, yet.

Opening and Saving Python Files

NOTE: Python files must have tabs as whitespace to be correctly processed correctly into the flow chart.

NOTE: FloPy DOES NOT ask to save your files when closing FloPy or create a new document, and file restore is not supported. Save Often.

To open a file, go to File→Open. This will open a file browser where you can find your .py file you wish to use.

To create a new file, go to File→New. This will clear all the flow chart components and editor. Make sure you save your work before selecting this.

To save a file, you can click either save or save as. Save will save you progress into the opened file. Save As will let you select a new file and location to save your .py file.

FloPy saves your code as a Python file. You can run the file just as any other python file in the Python Interpreter.

Conclusion

Thank you for using FloPy v0.1. I hope that you can bear with the kinks and me as I develop this program into something more than a class project.

I would like to give special thanks to:

- Microsoft and MSDN for the free Visual Studio access, along with the developer program subscription. MSDN was also a great resource for using the .NET framework in this program
- Various users on Stack Overflow for asking and answering questions, many of which I ran into while programming here.
- The IronPython team for making IronPython an open-source dotNET implementation of Python, allowing me to work with the Windows Presentation Foundation (WPF). WPF is originally coded for C#, hence the graphical kinks.
- **IRONPYTHON IN ACTION**, by Michael J. Foord and Christian Muirhead, for helping me quickly learn IronPython and WPF implementation.
- Dr. Susan Mengel and Mario Pitalua for being such great, helpful instructors in my first computer programming class after nearly 7 years of programming various languages.