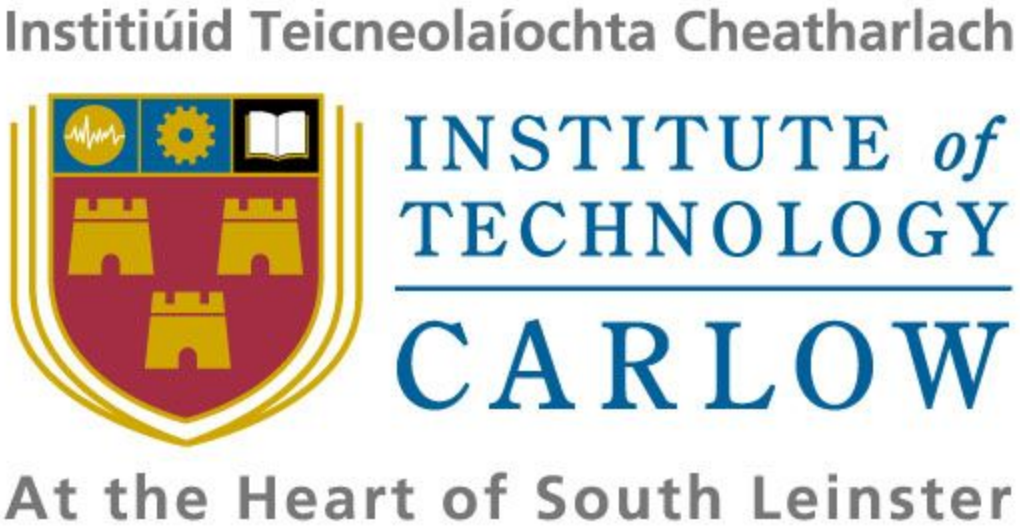


IT Carlow
Bachelor of Software Development

Year 4

Portable GUI for ptpython shell

Final Project Report

Student Name: **Inga Melkerte**
Student ID: **C00184799**
Supervisor: **Paul Barry**
Date: **04/04/17**

Table Of Contents

1. Introduction

This is final project report for the 4th year software development final project which took part from 29th of September 2016 until 4th of April 2017. The purpose of this document is to give an overview of this final project, called “Portable GUI for ptpython shell”.

Project consisted of 3 iterations:

1. 12 weeks
2. 7 weeks
3. 5 weeks

More information in the diary at the end of this report (see Appendix A).

This report is going to cover:

- description of the submitted project,
- description of conformance to specifications and design,
- description of learning, review of the report,
- diary of project,
- acknowledgments.

2. Description

2.1. Project proposal - Add portable GUI to ptpython shell.

Newcomers to Python are productive immediately thanks to the existence of the Python shell (>>>) and its GUI-equivalent, IDLE. Although both are usable tools, newbies quickly outgrow them (and they are both missing some tools/features which many programmers expect to be provided). As a result, a number of additional environments have been created to offer a more feature-full programming experience. Of all of them, ptpython (see:https://github.com/jonathanslenders/ptpython and https://pypi.org/project/ptpython/) is notable for provided many powerful features, while remaining easy-to-use. Unfortunately, ptpython works at the command-line only, and does not provide an IDLE-like GUI option. The goal of this project is to add a portable GUI to ptpython.

2.2. What is Ptpython?

Ptpython is developed by Jonathan Slenders, developer from Belgium. “Ptpython is an advanced Python REPL. It should work on all Python versions from 2.6 up to 3.5 and work cross platform (Linux, BSD, OS X and Windows).”(GitHub, 2017) Ptpython is built on top of prompt-toolkit library.

J.Slenders created prompt-toolkit library for building powerful interactive command lines and terminal applications in Python.

Ptpython provides great features such that:

- Syntax highlighting.
- Multiline editing (the up arrow works).
- Autocompletion.
- Mouse support.
- Support for color schemes.
- Support for bracketed paste .
- Both Vi and Emacs key bindings.
- Support for double width (Chinese) character, etc. (GitHub, 2017)

The code can be downloaded from https://github.com/jonathanslenders/ptpython.

2.3. PtpythonGui Description

PtpythonGui is being developed to provide Graphical User Interface for ptpython shell. This tool would be suitable for beginners as a learning tool. It would provide easy and user friendly interface with simple buttons and menu bar. It would offer ptpython cool features such as syntax highlighting, autocompletion , history search etc. This project is not finished because I run out of the time. I believe that I would be able to finish this project if I had about 4 more weeks.

The reason I choosed this project was to learn python programming language. And I enjoyed graphical programing in 2nd year when Java programing language using Swing API. So I wanted to do a graphical user interface in python. Actually it did not sounded that complicated just to add a layer of front end on an existing project. But it turned out very challenging, at some parts annoying, complicating, but also exciting and very much rewarding in fantastic learning experience.

2.4. Technologies Used

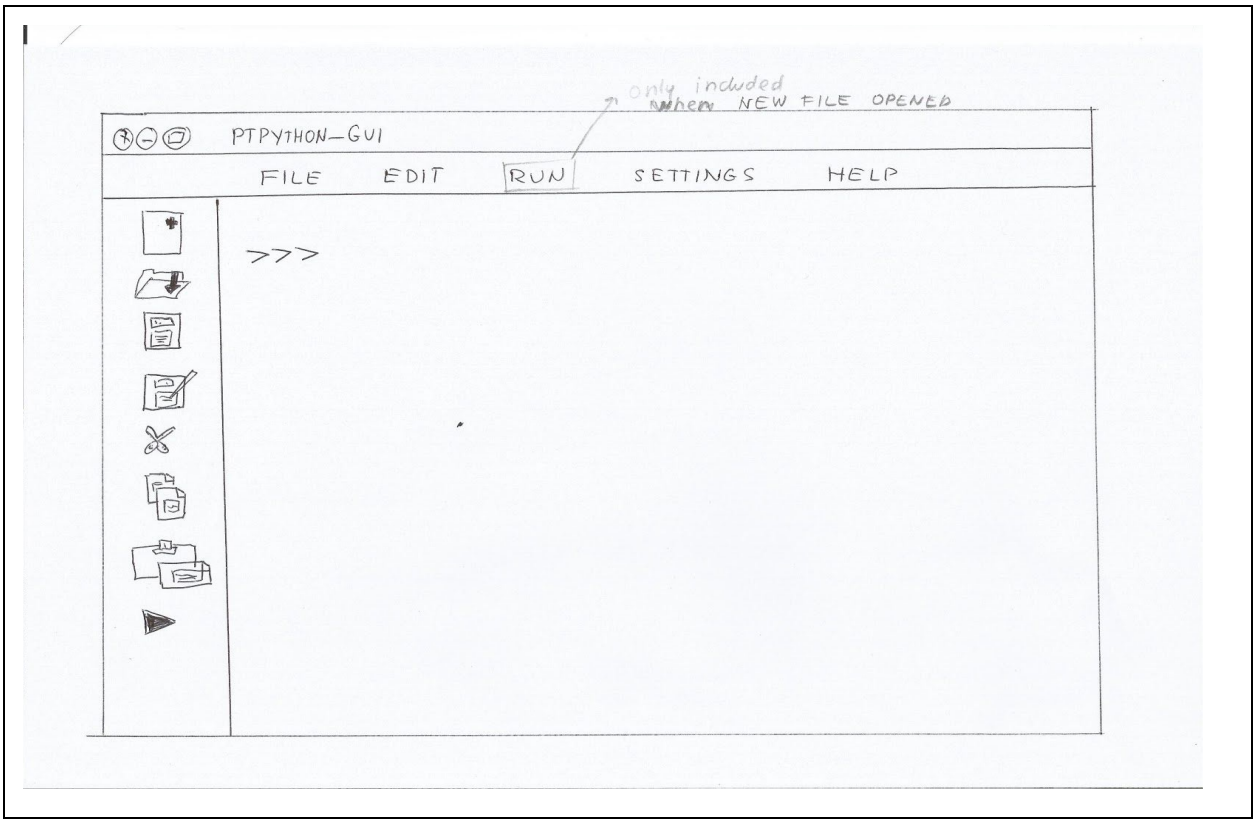
Programing language - python.
Researched GUI frameworks in python. For this project tkinter - GUI framework - was used to design and develop GUI application - PtpythonGui. The reason why Tkinter was chosen because it comes bundled up with python, it is included in Python standard library and there is no need to install it.There are lot of documentation and tutorials out there and Tkinter offers native look and feel on all platforms.
In the research phase - lots of GUI frameworks were researched and tested by creating simple application in each framework. GUI comparison table was created (it is included in research document).
Mercurial was used as version source control.
Virtual environment was created - VirutalEnv.

3. Description of Conformance to Specification and Design

The plan was to create PtpythonGui as user friendly, simple and intuitive IDE for beginners who wish to learn python programing language. It would be similar to IDLE but would offer some cool features what ptpython has such as syntax highlighting, autocompletion, search history, etc.

3.1. First iteration

Researched GUI frameworks in python (more in research document and also see Appendix A). Then researched about ptpython. Because I was new to python, I did not know python or ptpython. So I got familiar with ptpython and it's cool features. Then I wrote design and functionality specification documents for PtpythonGui. (see functional and design documents). First draft of the GUI was to have two windows - shell prompt window (start) and code editor window (when clicked on File -> New would open new window).



The GUI would have menu bar with buttons - File, Edit, Run, Settings and Help options. Right side toolbar would offer the shortcuts to some of options via graphical icons. First I had to identify entry point in ptpython (which I did - ptpython/entry_points/run_ptpython.py). And then I was going to bind “F3” function key to display history but this time not into the command line but into tkinter window. I could bind the “F3” function key to display new window but could not insert history information into GUI window. GUI actually changed and evolved as the project progressed.

3.2. Second iteration

I failed to implement simple prompt from prompt-toolkit library into GUI at the end of the first iteration and with the help of my supervisor Paul Barry I changed strategy and researched on how does REPL works in python and how to implement my own REPL. The strategy was changed because Ptpython is better python REPL - basically I need to know how python REPL works to understand ptpython REPL.

3.2.1. REPL in command-line

A Read–Eval–Print Loop (REPL), also known as an interactive toplevel or language shell, is a simple, interactive computer programming environment that takes user inputs, evaluates them, and returns the result to the user (Wikipedia, 2017).

“In some programming languages, eval is a function which evaluates a string as though it were an expression and returns a result; in others, it executes multiple lines of code as though they had been included instead of the line including the eval. The input to eval is not necessarily a string; it may be structured representation of code, such as an abstract syntax tree (like Lisp forms), or of special type such as code (as in Python). The analog for a statement is exec, which executes a string (or code in other format) as if it were a statement; in some languages, such as Python, both are present, while in other languages only one of either eval or exec is. “(Wikipedia, 2017)

In python 2 raw_input function was for getting input from user. The function was renamed for Python 3.x from raw_input() to input(). It returns string but lot of things are happening in the background.

Below is code of a very simple REPL in command-line I build - asks for user input (READ), if it is single statement it uses eval() function to evaluate statement (EVAL),

otherwise it is an expression (multiline statement) and it uses exec() function to execute this expression (EXEC), it prints the output (PRINT) and then it loops back using while loop (LOOP).

```
def main():
    namespace = {}
    while True:
        try:
            # single-line statements only
            line = input('>>> ')
        except EOFError:
            pass
        else:
            # first eval()
            temp = eval(line)
            if temp:
                print(temp)
            else:
                # or exec ()and print result
                exec(line, namespace)
if __name__ == '__main__':
    main()
```

```
inga@inga-SATELLITE-C855-1J2:~/D
e$ python cmd.py
>>> 9*9
81
>>> print("hello, world!")
hello, world!
```

Output of REPL in command line.

3.2.2. REPL redirected to GUI

First I implemented my own python REPL in command-line and then I created simple GUI for it.

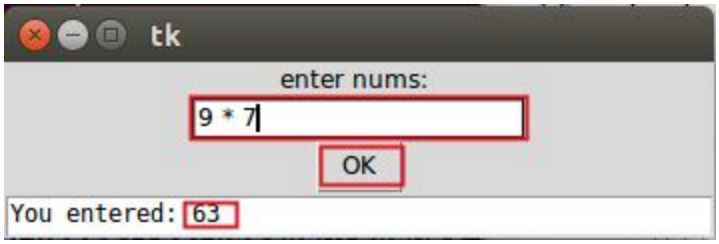
Steps taken to create my own python REPL in GUI:

- Redirected output from the command line to GUI tkinter window textbox (by overwriting write function instead of displaying output in command-line, it outputs text into output text area, which I disable to protect it from users changing it and first I need to change state to normal , so I can display output and then disable it again)When print function is called, it is redirected to output frame by overwriting write function.

```
#for textbox
def write():
    output.insert(END, str(txt))

def write(self, txt):
    #enable text area to output result
    self.output.configure(state="normal")
    self.output.insert(END,str(txt))
    #disable again
    self.output.configure(state="disable")
```

➢ Redirected user entered input - allow user to type input into tkinter entry box

➢ Binded Run button to functions eval() and exec()



➢ Added tkinter textbox and implemented multi-line editing for user input

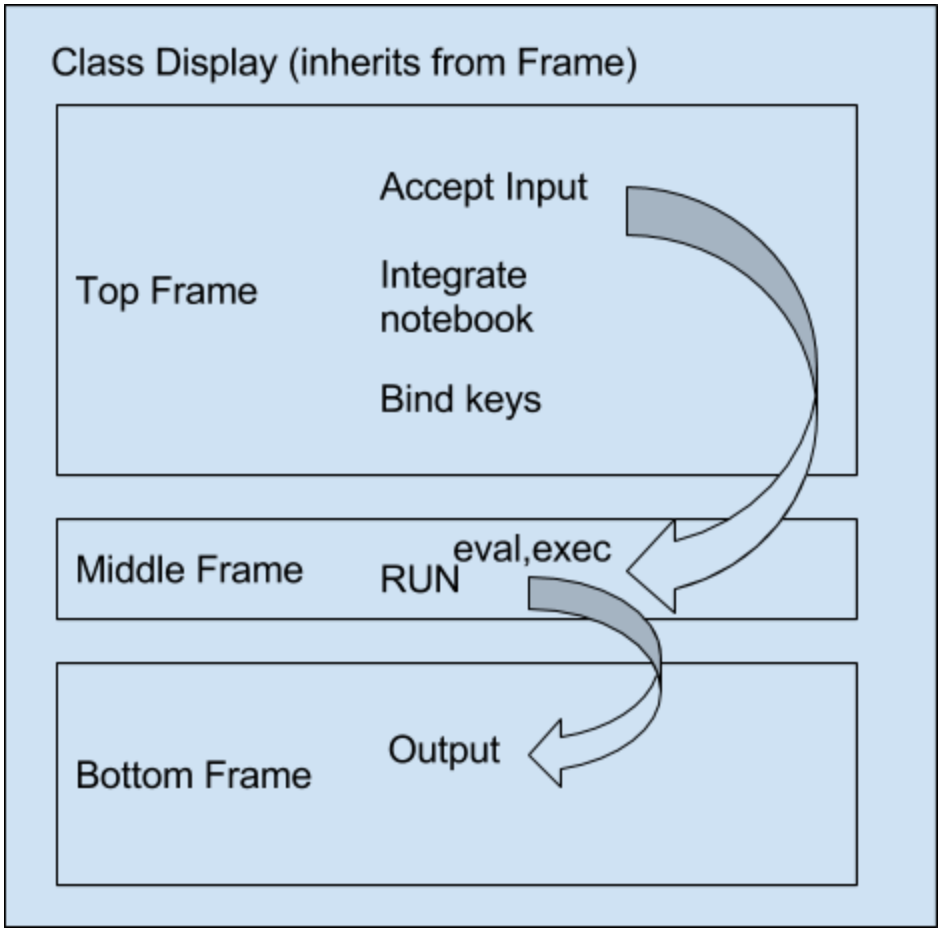
- Added 3 frame layers to GUI -
  - Top - frame (integrated notebook with tabs, scroll-bar and line-numbers)
  - Middle - frame (for buttons - run, open, save, clear screen, new tab)
  - Bottom - frame (for redirected output which is disabled, includes error redirection)

```
# top for notebook with text and scrollbar
self.top = Frame(self)

# middle for buttons
self.middle = Frame(self)

# bottom for output text
self.bottom = Frame(self)

self.top.pack(side = TOP)
self.middle.pack(pady= 15)
self.bottom.pack(side = BOTTOM, fill=BOTH, expand =True)
```

Top frame contains - shell window at the top of the application,which accepts user input

Middle - there are buttons - Run, Open, Save, Clear Screen, New. Run button evaluates or executes the code

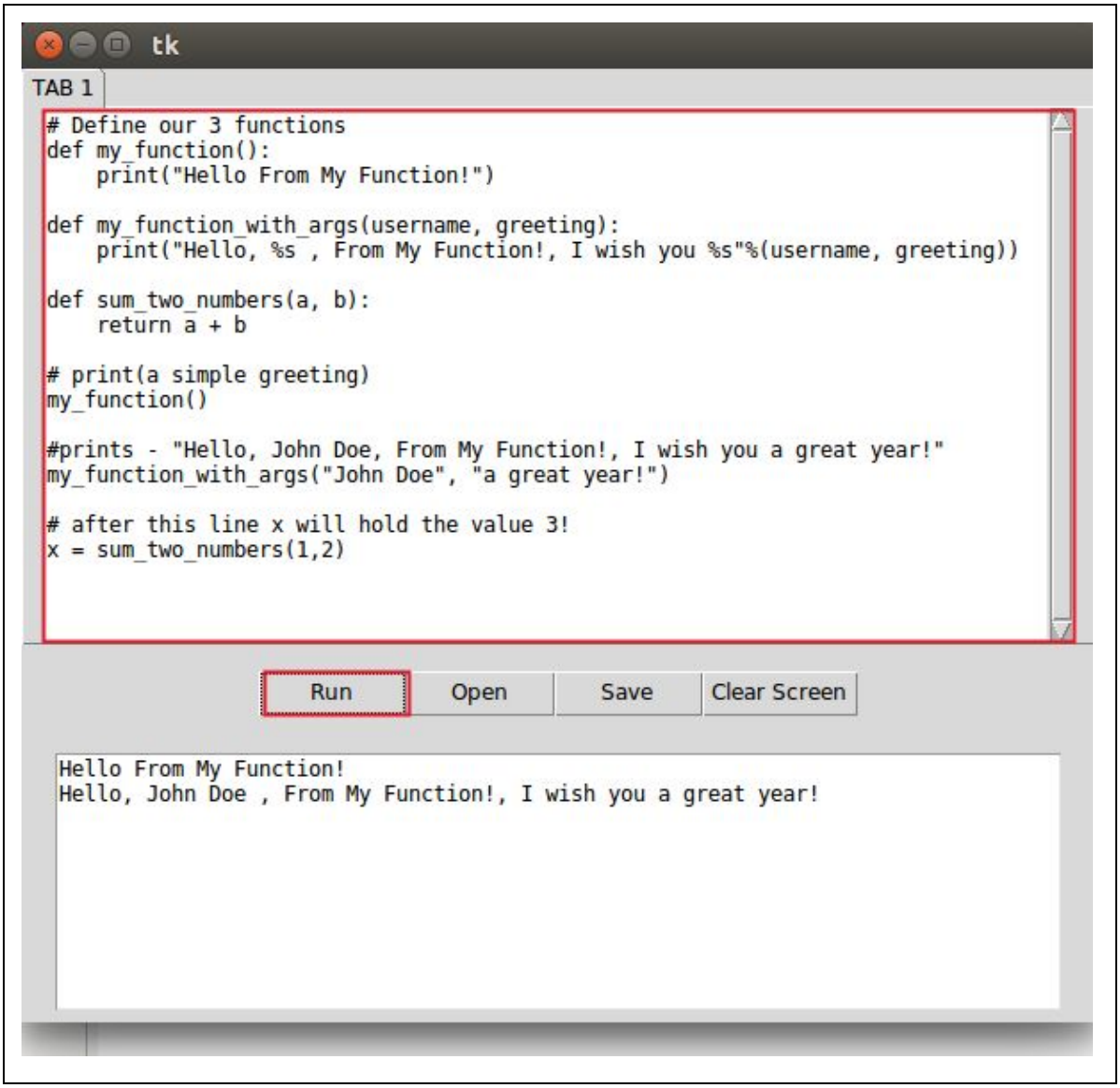
Bottom - at the bottom part there are text area for displaying (redirects) output.



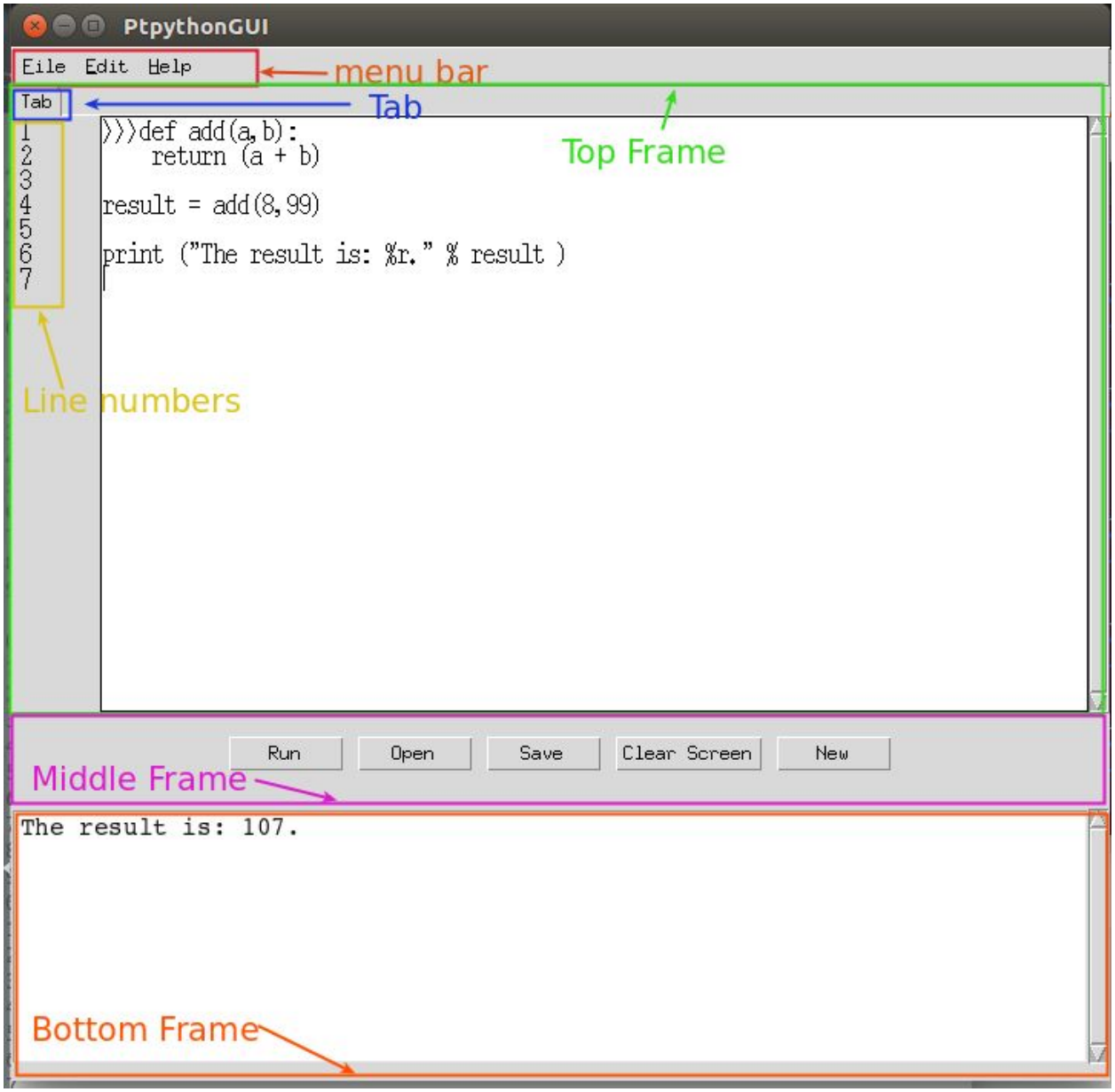
High level description of GUI

- Error handling - redirected errors into tkinter output

```
#assign sys.exc_info() to access (type, value, traceback).
error = sys.exc_info()
print(error[0],error[1])
```



For this part I used The tkinter.ttk module which provides access to the Tk themed widget set. The Ttk Notebook widget was used to implement tab and scrolled bar.



Functionality has changed - it is very simplified IDE version, just to give user ability to run code by pressing button Run, access files from the local computer via Open button and display them in main text area, Save button gives option to save the file, Clear Screen button - removes text from textarea fast and easy, button New gives option to open a new tab.

I changed the idea of using notebook from ttk widgets because I found already made Notebook in github and I integrated this notebook from github - which has tab and binds line numbers as enter key is being pressed. It is a nice feature to be add to IDE because for example IDLE does not have line numbers at the sidebar. But line numbers

gives indication for the user when error occurs. When Error occurs it would display at the bottom frame textarea and tell on which line.

3.2.3. Ptptython REPL into GUI

> *"It is easier to write an incorrect program than understand a correct one."*
> Alan J. Perlis (Perlis, 1982).

Difficulty to understand and trace the ptpython code and prompt-toolkit. Jonathan Slenders is fantastic programer I have learned so much by reading his code but he is also using all the "tricks" that I am not familiar with. Code is complicated. Just to show that there are lots of code here are Ptpython and prompt-toolkit library files and lines of code:

Ptpython - has 20 files

- -contrib
  - -__pycahce__
  - -asyncssh_repl.py
  - -__init__.py
- -entry_points
  - -__pycahce__
  - -__init__.py
  - -run_runptipython.py
  - -run_ptpython.py
- -_pycache_
  - -completer.py (153)
  - -custorepl.py (51)
  - -_eval.py (16)
  - -eventloop.py (75)
  - -filters.py (37)
  - -history_bowser.py (596)
  - -__init__.py
  - -ipython.py (302)
  - -key_bindings.py (235)
  - -layout.py (573)
  - -__main__.py (7)
  - -prompt_style.py (77)
  - -python_input.py (672)

-repl.py (302)
-run_ptpython.py (75)
-style.py (195)
-utils.py (124)
-validator.py (44)

PROMPT-TOOLKIT has 104 files (some of them are listed down below)

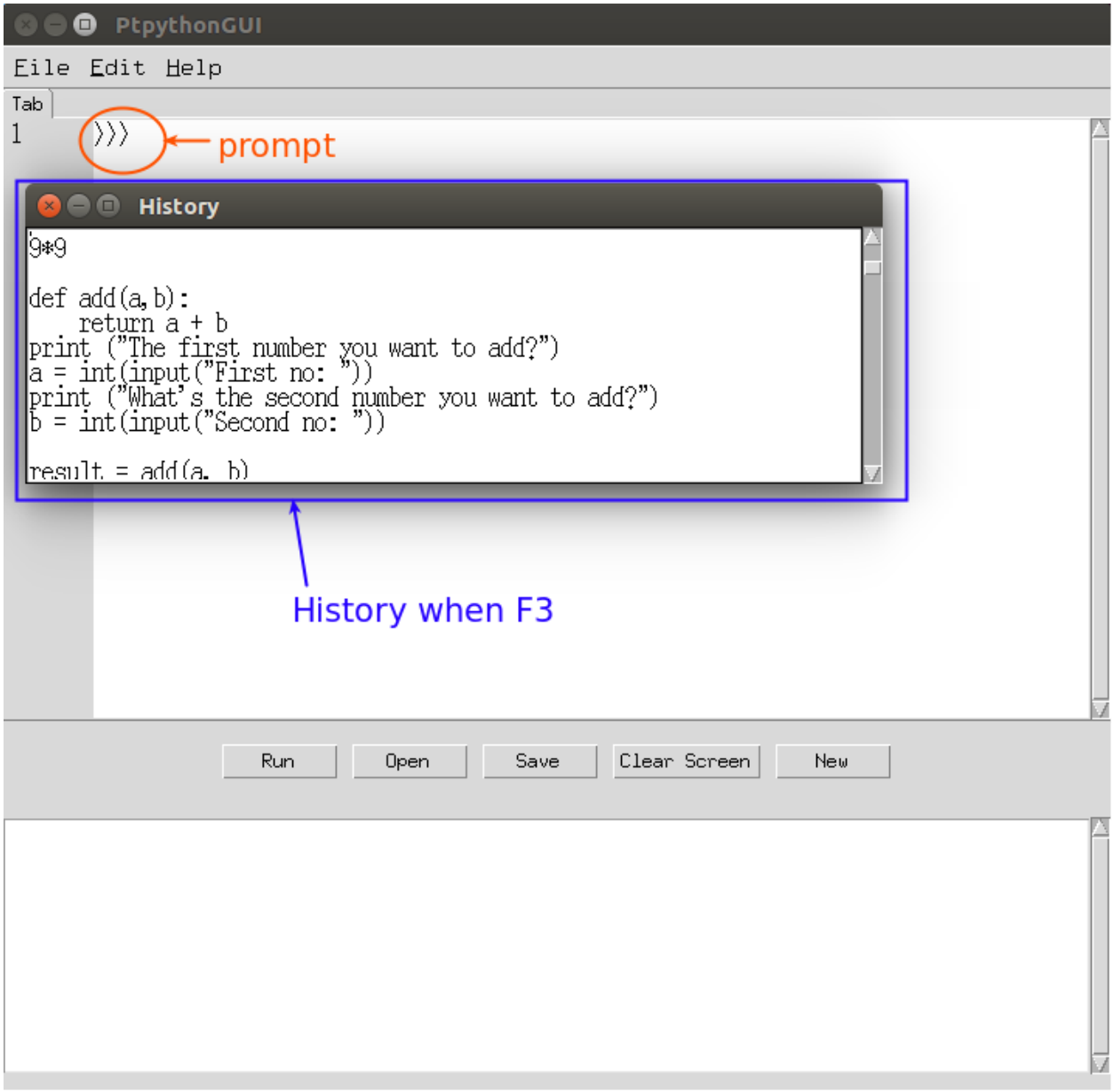
-clipboard
-contrib
-eventloop
-filters
-key_binding
-layout
-containers.py (1665)
-controls.py (730)
-dimensions.py (92)
-__init__.py
-lexers.py (320)
-margins.py (253)
-menus.py (496)
-mouse_handlers.py (29)
-processors.py (605)
-prompt.py (111)
-screen.py (151)
-toolbars.py (209)
-utils.py (181)
-__pycache__
-styles
-base.py (86)
-defaults.py (95)
-from_dict.py (148)
-from_pygments (77)
-__init__.py
-utils.py (45)
-terminal
-application.py (192)
-auto_suggest.py (88)

-buffer.py (1415)
-buffer_mapping.py (92)
-cache.py (111)
-completion.py (170)
-document.py (1001)
-enums.py (29)
-history.py (120)
-__init__.py (22)
-input.py (135)
-interface.py (1185)
-keys.py (129)
-mouse_events.py (48)
-output.py (192)
-reactive.py (56)
-renderer.py (526)
-search_state.py (36)
-selection.py (47)
-shortcuts.py (717)
-token.py (47)
-utils.py (240)
-validation.py (64)
-win32_types.py (155)

3.3. Third iteration

The third iteration I am understanding more about REPL and how ptpython REPL works.

- Traced back code
- Implemented prompt (>>>)
- Binded history display on F3 key



- Binded Up-Arrow key to display autocompletion
- Added menu bar with menu options
    i. File - Run, Open, Save, Clear Screen, New
    ii. Edit - Cut, Paste
    iii. Help - About

4. Description of Learning

4.1. Technical Learning

4.1.1. Graphical programing

I researched lots of different GUI frameworks in python. For this project Tkinter was chosen - because it is inbuilt in python and it provides lots of documentation and tutorials. Python is a new programing language for me and I had to learn graphical programming in python. It is actually much easier to write code for GUI in python than in for example Java. I am delighted I had opportunity to learn this awesome language which I hope I will use in future.
I got insight in lots of python GUI frameworks and I had an opportunity to create a simple examples in them and then compare them based on its features.

4.1.2. Build your own REPL

I learned how to build my own python shell and REPL, and then implement it into GUI.
I learned lots from reading code, tracing back code by hand and in debugger.

4.1.3. Understand someone else's code

J.Slenders is fantastic programer and he uses all the “tricks” to write prompt-toolkit and ptpython. This ability to read somebodies else's code will be advantage when I will start my “real work” in “real world”.

4.1.4. Integrate already existing code

Another learning outcome - Integrate code - I had to integrate notebook with line numbers into my code and that is important skill to know. And manipulate it. First I was going to use tkk widget notebook but it was really much work to implement line numbers and tabs. So when I found already made notebook I reused it in my code because it save me time.

4.2. Personal Learning

4.2.1.Communications

Communication is very important skill. I learned to communicate ideas effectively with my supervisor. I had to go the the meetings and admit that I could not achieve what I had planned but there were always different strategy for a next week.

4.2.2. Presentation

Presentation skills are important key of getting message across by presenting information clearly and effectively and these skills are important in software development area as well. I learned that from the work placement experience last year. So I was delighted to have another chance to improve my presentation skills. I have learned a lot over 4 years in college. I still have to work on building my confidence. I get very scared and stressed doing presentations but I think that the more I practice the “easier” it gets.

4.2.3. Work independently

This is first time I was working on the project for so long period on my own. This was individual project but I had improve on multitasking skills - final project was ongoing from september till april but during this time we had other classes, continues assessments and other project deadlines.

4.2.4. Listen to feedback

I attended all my assign meetings with my supervisor because I feel very responsible and I valued the time I had with my supervisor as great learning experience: I listened to his feedback on the work I had done each week, I took his advice for next week.

4.2.5. Problem solving

Problem solving is part of software development process. During the project time so many problems occurred but most importantly I always tried to solve them by myself and there were many times I had to ask my supervisor Paul Barry for help. And I learned so much by solving my problems by myself but it was unbelievable how much I learned just by watching my supervisor to fix the problems I felt privileged to have such an amazing supervisor.

4.2.6. Time management

I was working really hard this year and I had to manage my time. I used a diary and I planned only week ahead for my project. It was very overwhelming to think of everything altogether so I learned to take day by day and take simple steps, to break down work in little tasks.

5. Review of Project

5.1. What went wrong?

During this project there were lots of challenges and things did go wrong. At the beginning of the project during the first iteration when I researched about GUI frameworks in python - lots of times there were problems in installing them. I was using laptop with ubuntu and python is installed in ubuntu but I had python 2.7 version. When I installed python 3.5 and some tutorials would provide information about tools only for python 2.7.

Often I felt overwhelmed with the ptpython and prompt-toolkit code. The more I looked, the more I found that there is more code. I was thinking about it, dreaming about it. It was great idea to take a step back at the beginning of the second iteration and change strategy to build REPL because I took a "break" from trying to understand code and focused on different tasks and when I looked back at the code I could understand it more and identify points I did not spot before.

J. Slenders provided documentation for prompt-toolkits on March 02, 2017 (I found it a bit later but it was already middle of second iteration) but that was great help. When I started to read the documentation I started to make progress but it was a bit too late for this project.

5.2. What went right?

At the beginning of the project I had no idea how to approach this project, what steps to take. It was very challenging project for me. The most importantly I learn so much during the project. I chosed this project to learn python programing language and especially graphical programming in python. I don't have much experience in programing and python was new language to me. But I absolutely love python and I am delighted I chose this project. 2 iterations went so quick and I was overwhelmed with complicity of already existing code from ptpython and prompt-toolkits.Third iteration was much shorted but suddenly something "clicked" and I was able to bind F3 key to display history from prompt function and bind autocomplete to up-arrow key.

I believe this was very difficult project for me but at the end of the third iteration when I am able to implement prompt features into my GUI I felt very excited and happy and I wish I had a little more time to complete my project but it gives me feeling of a little success.

5.3. What is still outstanding?

There are still lots of work outstanding to finish this project "Portable GUI for ptpython shell". I have implemented few ptpython features but there are lot more ptpython features to be implemented. Then I wanted to improve on GUI design, add more menu bars, buttons and icons. I would have wanted to test it more and get feedback from users.

5.4. What would I do differently?

I would not spent too much time on researching on GUI frameworks in python. I was worried about graphical programing but it is much easier and fun to program in python that in Java. I run in lots of problems when trying out GUI frameworks - some were difficult to install and compliance with python 2.7 and python 3.5 was annoying.
I would faster move to develop simple REPL in python because it gave me understanding how REPL works and how ptpython works and then I would look at prompt-toolkit library.

5.5. Advice for similar projects

This project would be easier done for person who has experience with python language, understands what is REPL and has some experience in graphical programing.

Spend less time in researching GUI frameworks, choose at the start one GUI tool and start to build GUI for python REPL and then extend it by adding prompt() function parameters. And work in agile style with completing simple steps as the time progresses - write simple set of requirements, it's functionality, design and the write the code for it . And keep in mind end user - person who wish to learn python language - and find some of them to give them to try out the product and get the feedback fast.

6. Acknowledgments

I would like to express my special thanks to my supervisor Paul Barry - he is an amazing programer and a fantastic lecturer. I have learned so much from him during my project time. I felt privileged to have him as my supervisor.
During the times when I felt overwhelmed and lost with the project (believe me - there were lots of them) - I could always reach out - email him and get straight away reply and he was always available during the class or at our meetings to support, to help, to advise and to point into the right direction.
He is an inspiration and I am extremely thankful to Paul for his patient, encouragement and advice what he has provided to me through this project.

Appendix A

Diary

1st Iteration

Week	Date	Task	Action
1.	29/09/16	Topic chosen - Portable GUI for ptpython shell	
2.	5/10/16	Install and “play” with ptpython	Installed ptpython
3.	12/10/16	Research and compare GUI tool kits in python (see Appendix A)	Some errors using ptpyton - because it was running in python2.7.Paul helped me to fix it for me. :)
4.	19/10/16	Design simple GUI application using each GUI tool and record how easy was to install them, any difficulties.	1.**Tkinter** - easy enough to find inf.and design sample app. 2.**Toga** - more challenging, very little inf., when running sample code, I got errors, trying to fix it and get it working 3.**PyQt**- https://docs.google.com/document/d/15dY2nUomC_cVhgnzIfBs9BEgnk44VJBDxMU0T5dYMSc/edit 4.**EasyGui -** Not-event driven, invoked by simple function calls, don’t need to know about tkinter, frame, widgets, callbacks, lambda.Uses tkinter back-end 5.**PySide -** Paul helped me to fix problems with downloading GUI frameworks and indentation problem.
5.	26/10/16	Research further GUI toolkits. Create simple application in dif. GUI’s	**PyQt** - ex8.py - input, label ,button **Kivy -** researched more and I found out that it is more for mobile app dev.- supports multi-touch

			EasyGui-
6.	2/11/16	Start to write research manual and functional specifications. Research can easygui and tkinter can be used interchangeably	1. Started to write documents ● Research Manual ● Functional Specification 2. Easygui and tkinter can be used interchangeably - Easygui ex2.py 3. Looked at ptpython code and when I type ptpython I can output window 4. Read about prompt-toolkits 5.Listened to podcast interview with J.Slenderes about Prompt-toolkit https://www.podcastinit.com/episode-6-jonathan-slenders-talks-about-prompt-toolkit/
7.	9/11/16	Look at ptpython code Chose Tkinter/EasyGUI	Window ptpython
8.	10/11/16	Display window when F3 pressed	1.I can display window when i run run_ptpython.py 2.updated research document 3.create use cases
9.	17/11/16	Download VirutalEnv for python, learn how to use it	Virtual environment downloaded, Still ptpython is trying to reach out of virtual environment
10.	24/11/16	VirutalEnv problems	Accessing everything outside virtualenv, weird (Paul helped me to fix it)
11.	1/12/16	Chose Mercurial as source control tool	Followed tutorials and pushed all the code to bitbucket using mercurial
12.	8/12/16	Preparation for presentation	Draft created for presentation
13.	13/12/16	1ST PRESENTATION	Presentation Done

2nd Iteration

Week	Date	Task	Action
1.	18/01/17	Task - implement prompt into GUI	Not successful, cannot identify the starting point
2.	25/01/17	Research on REPL for python	Paul sent me links, I found some too - following tutorials to build simple python REPL, cmd and YOSH.
3.	1/02/17	Simple REPL in command line build. Wrap GUI around REPL	Input entered into Tkinter entry box, (reads, evaluates or executes, prints, loops back), and output is displayed in TKinter label
4.	8/02/17	Add text box for multi line editing	Textbox allows multiline editing and exec function does it.
5.	15/02/17	Exceptions handle and display them into GUI instead of command line	Done
6.	22/02/17	Add notebook into GUI	Added notebook with line numbers and tabs into my GUI
7.	28/02/17	2ND PRESENTATION	Done

3rd Iteration

Week	Date	Task	Action
1.	7/03/17	Back looking at ptpython and prompt-toolkit library	J.Slenders have put up prompt-toolkit documentation, reading that, I have identified where ptpython evaluates code but I have to trace back more
2.	14/03/17	Tracing back through ptpython and prompt-toolkit (buffer, cli, input, prompt, repl)	Prompt-toolkit library is complex. Writing final project report
3.	21/03/17	Embedded prompt (>>>) Implementing history binding to F3 key	Hidden history file is been created. I can display history in TKinter Window when F3 is pressed
4.	28/03/17	Bind Up-Arrow key to autocompletion Updating Final Document	I have binded Up-Arrow to simple autocompletion Final Document done
5.	4/04/17	Demo - Final project Deadline	Demo

References

1. GitHub. (2017). jonathanslenders (Jonathan Slenders) · GitHub. [ONLINE] Available at: https://github.com/jonathanslenders. [Accessed 17 March 2017].

2. Perlis, A. J. (September 1982). "Epigrams on programming”

3. Slenders, (2017). prompt-toolkit documentation. [ONLINE] Available at: https://media.readthedocs.org/pdf/python-prompt-toolkit/latest/python-prompt-toolkit.pdf [Accessed 30 March 2017].

4. Wikipedia (2017). [ONLINE] Available at: https://en.wikipedia.org/wiki/GNU_Readline [Accessed 17 February 2017].