

Institiúid Teicneolaíochta Cheatharlach



INSTITUTE *of*  
TECHNOLOGY  

---

CARLOW

At the Heart of South Leinster

IT Carlow

Bachelor of Software Development

Year 4

# Portable GUI for ptpython shell

Design Document

Student Name: **Inga Melkerte**

Student ID: **C00184799**

Supervisor: **Paul Barry**

Date: **14/11/16 - 04/04/17**

## Table of Contents

### **1. Introduction**

1.1. Project overview

### **2. Project Plan**

2.1. First Iteration

2.1.1. Embed ptython

2.1.2. Appendix A

2.2. Second Iteration

2.2.1. prompt-toolkit library

2.2.2. Appendix B

2.3. Third Iteration

2.3.1. Ptython and prompt-toolkits

2.3.2. Appendix C

2.3.3. Examples from prompt-toolkit

### **3. Reference**

# 1. Introduction

The purpose of this design document is to outline the technical design of the final year project “Portable GUI for ptython shell”.

## 1.1. Project overview

This project aims to develop a portable GUI for ptython shell as an IDE for beginners who wish to learn python programming language. It is going to be build on top of already existing command-line interface - ptython. Here is the strategy and steps taken through the project to understand ptython and prompt-toolkits. Some of the code from these libraries are included in this document. This design document has been updated as the project progressed throughout iterations.

## 2. Project Plan

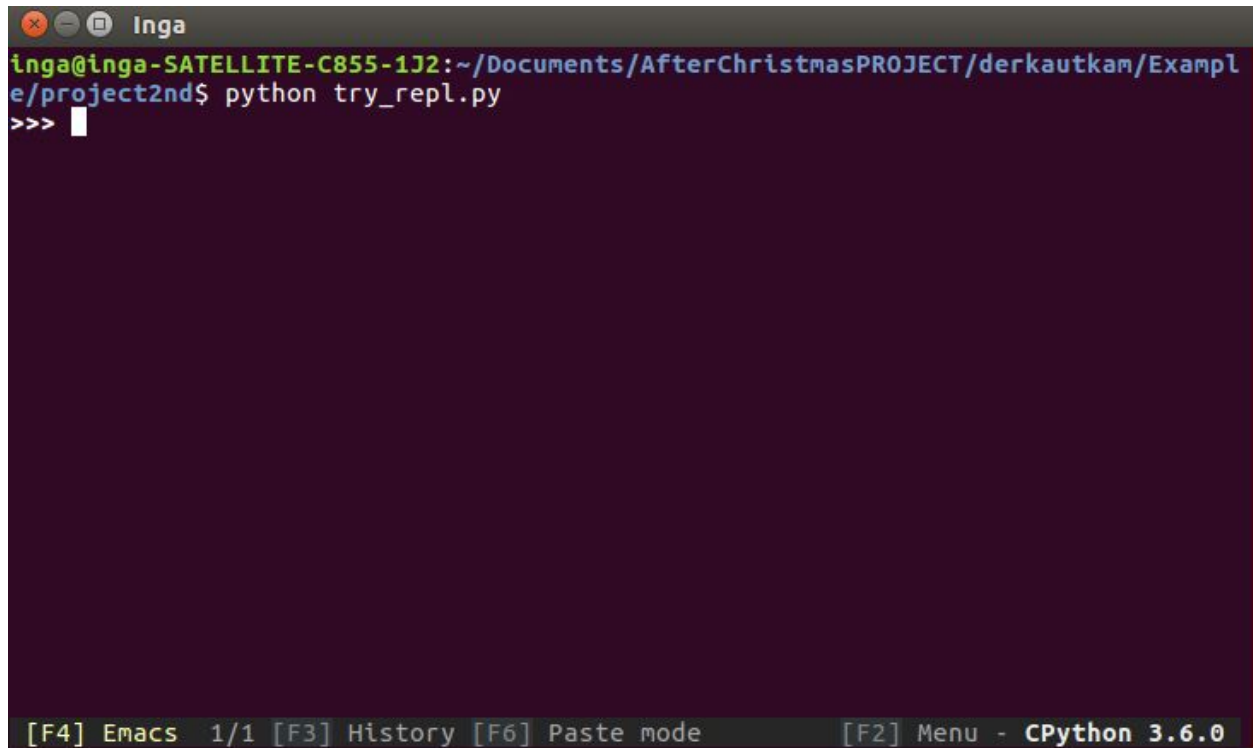
### 2.1. First Iteration

- research python GUI frameworks
- get familiar with ptython

#### 2.1.1. Embed ptython

Embedding the ptython REPL in any Python application (terminal ) is easy:

```
1 from ptython.repl import embed
2 embed(globals(), locals(), title = "Inga", vi_mode=False)
```

A terminal window titled 'Inga' with a dark purple background. The prompt is 'Inga@Inga-SATELLITE-C855-1J2:~/Documents/AfterChristmasPROJECT/derkautkam/Example/project2nd\$'. The command 'python try\_repl.py' has been executed, resulting in a Python REPL prompt '>>>' with a white cursor. The status bar at the bottom shows '[F4] Emacs 1/1 [F3] History [F6] Paste mode [F2] Menu - CPython 3.6.0'.

```
Inga@Inga-SATELLITE-C855-1J2:~/Documents/AfterChristmasPROJECT/derkautkam/Example/project2nd$ python try_repl.py
>>> 
```

Identify entry point into ptython

(*ptython/ptython/entry\_points/run\_ptpython.py*)

Display window when F3 pressed (bind it to display history\_browser)

(*ptython/ptython/history\_browser.py*)

## 2.2. Second Iteration

### 2.2.1. prompt-toolkit library

**prompt ()** function is found in

`python-prompt-toolkit/prompt_toolkit/shortcuts.py` file

Prompt function (which is the main function like `input` in python) offers all the cool `ptpython` features. J.Slenders comments that prompt function is “to get input from the user and return it.” He also says - “ this is a wrapper around a lot of “prompt-toolkit” functionality and can be a replacement for “`raw_input`”. (or GNU `readline`)”

(Slenders,2017)

Simple python `input()` function

```
text = input("Give me some input: ")
print("You said: %s" % text)
```

Simple `ptpython` `prompt()` function

```
text = prompt("Give me some input: ")
print("You said: %s" % text)
```

These two simple examples shows that `prompt()` function in `ptpython` is just like (`raw_input()`) or `input()` functions in python. It asks for user input and returns the text.

This `prompt()` function is so powerful, it has lots of parameters. Prompt can be easily embedded into terminal applications and it has lots of configuration options which are provided through these parameters - history, autocompletion, syntax highlighting, he actually uses `pygments` library to achieve syntax highlighting, but it is built into `prompt` function.

### **create\_prompt\_layout**

It is a container instance for `prompt`. This function draws UI in command-line interface. It imports `layout/containers.py` - creates container, splits command-line interface horizontally using `HSplit` and vertically - `VSplit`.

`(python-prompt-toolkit/prompt_toolkit/shortcuts.py)`

Layout of the terminal application is created in

```
python-prompt-toolkit/prompt_toolkit/layout/containers.py
```

It creates **Container**, **HSplit**, **VSplit**, **FloatContainer**, **Float**, **Window** and more. It is very large file and it draws the UI for command-line interface.

```
python-prompt-toolkit/prompt_toolkit/interface.py
```

**CommandLineInterface** (CLI) class is created which is a wrapper around all the other classes, tying everything together.

Typical usage::

```
application = Application(...)
cli = CommandLineInterface(application, eventloop)
result = cli.run()
print(result)
```

**run\_system\_command()** function waits and listens for enter key being pressed - enter key is binded to key J, key M and also Enter key.

## 2.3. Third Iteration

### 2.3.1. Ptpython and prompt-toolkits

`python-prompt-toolkit/prompt_toolkit/buffer.py`

In this file data structures for buffer are included. It holds text, cursor position, history, etc.

Buffer is the core data structure that holds the text and cursor position of the current input line and implements all the text manipulations on top of it. It also implements the history, undo stack and completion state (Slenders, 2017). It has lots of attributes - completer, auto\_suggest, history, validator, is\_multiline, complete\_while\_typing, etc.

`ptpython/ptpython/repl.py`

#### **repl()**

PythonRepl class - is subclass of PythonInput Class

Process\_document has parameters- cli and buffer

Execute - function where REPL is created. I had to back trace buffer

Embed - very powerful function - can be called at any point in newly design program to embed ptpython

`ptpython/ptpython/python_input.py`

Python\_input creates 2 classes - PythonInput and PythonCommandLineInterface .

PythonInput is for prompting reader for python input.

`ptpython/ptpython/prompt_style.py`

Prompt\_style in ptpython styles prompt, it contains base class for all the prompts and abstract methods. Can choose from lpython prompt and classic prompt.

`ptpython/ptpython/layout.py`

Layout creates and draws layout UI for ptpython shell in command-line.

### 2.3.2. Examples from prompt-toolkit

These are examples taken from prompt-toolkit library. They were used to implement them into GUI.

```
from __future__ import unicode_literals
from prompt_toolkit import prompt
from prompt_toolkit.history import InMemoryHistory

def main():
    history = InMemoryHistory()

    while True:
        text = prompt("> ", history=history)
        print('You entered:', text)
        print('GoodBye!')

if __name__ == '__main__':
    main()
```

(GitHub, 2017).



```
#!/usr/bin/env python
"""
Autocompletion example.
Press [Tab] to complete the current word.
- The first Tab press fills in the common part of all completions
  and shows all the completions. (In the menu)
- Any following tab press cycles through all the possible completions.
"""
from __future__ import unicode_literals

from prompt_toolkit.contrib.completers import WordCompleter
from prompt_toolkit import prompt

animal_completer = WordCompleter([
    'alligator',
    'ant',
    'ape',
    'bat',
    'bear',
    'beaver',
    'bee'
], ignore_case=True)

def main():
    text = prompt('Give some animals: ', completer=animal_completer,
                 complete_while_typing=False)
    print('You said: %s' % text)

if __name__ == '__main__':
    main()
```

(GitHub, 2017).

## 3. Reference

1. GitHub, (2017). python-prompt-toolkit/examples/tutorial at master · jonathanslenders/python-prompt-toolkit · GitHub. [ONLINE] Available at: <https://github.com/jonathanslenders/python-prompt-toolkit/tree/master/examples/tutorial>. [Accessed 04 March 2017].
2. 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].