

```
time import datetime
time import d
s_minute = da date
,3,5,7,9,11,1 datetime
7,49,51,53,55 datetime
nt_this_minute)
this_minute in odds:
("That number is a li
("But it is not weird
("Not an odd minute."
("It is even.")
("We are done")
```

# PORTABLE GUI FOR PTPYTHON SHELL

Student Name: **Inga Melkerte**

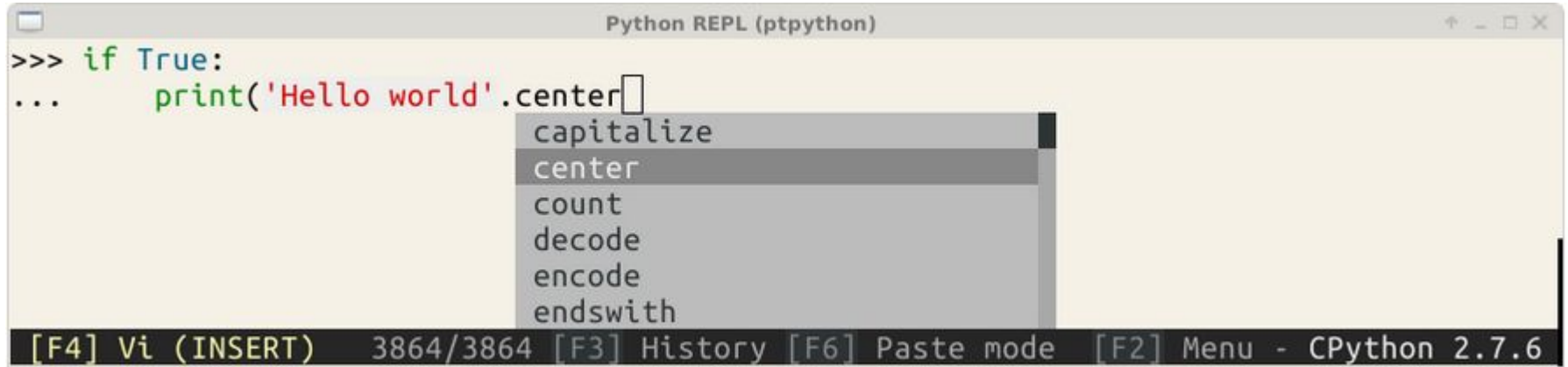
Student ID: **C00184799**

Supervisor: **Paul Barry**

Date: **28th February 2017**

## Reminder - what the project is about?

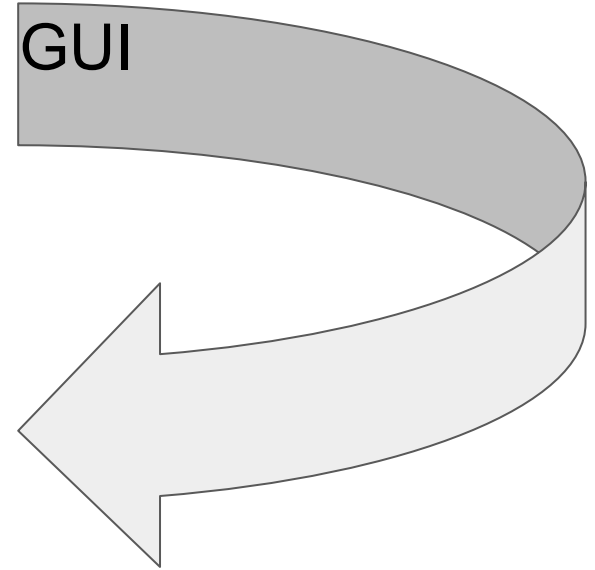
- ❖ Build GUI framework for ptpython shell
- ❖ Similar to IDLE
- ❖ Ptpython cool features



```
Python REPL (ptpython)
>>> if True:
...     print('Hello world'.center
          capitalize
          center
          count
          decode
          encode
          endswith
[F4] Vi (INSERT) 3864/3864 [F3] History [F6] Paste mode [F2] Menu - CPython 2.7.6
```

## Plan from the last iteration

Implement prompt from prompt-toolkit into GUI



```
22 from __future__ import unicode_literals
23
24 from .buffer import Buffer, AcceptAction
25 from .document import Document
26 from .enums import DEFAULT_BUFFER, SEARCH_BUFFER, EditingMode
27 from .filters import IsDone, HasFocus, RendererHeightIsKnown, to_simple_filter, to_cli_filter, Condition
28 from .history import InMemoryHistory
29 from .interface import CommandLineInterface, Application, AbortAction
30 from .key_binding.defaults import load_key_bindings_for_prompt
31 from .key_binding.registry import Registry
32 from .keys import Keys
33 from .layout import Window, HSplit, FloatContainer, Float
34 from .layout.containers import ConditionalContainer
35 from .layout.controls import BufferControl, TokenListControl
36 from .layout.dimension import LayoutDimension
37 from .layout.lexers import PygmentsLexer
38 from .layout.margins import PromptMargin, ConditionalMargin
39 from .layout.menus import CompletionsMenu, MultiColumnCompletionsMenu
40 from .layout.processors import PasswordProcessor, ConditionalProcessor, AppendAutoSuggestion, HighlightSearchProcessor, HighlightSe
41 from .layout.prompt import DefaultPrompt
42 from .layout.screen import Char
43 from .layout.toolbars import ValidationToolbar, SystemToolbar, ArgToolbar, SearchToolbar
44 from .layout.utils import explode_tokens
45 from .renderer import print_tokens as renderer_print_tokens
46 from .styles import DEFAULT_STYLE, Style, style_from_dict
47 from .token import Token
48 from .utils import is_conemu_ansi, is_windows, DummyContext
49
50 from six import text_type, exec_, PY2
51
52 import os
```

```

1 def run_application(
2     application, patch_stdout=False, return_asyncio_coroutine=False,
3     true_color=False, refresh_interval=0, eventloop=None):
4     """
5     Run a prompt toolkit application.
6     :param patch_stdout: Replace `sys.stdout` by a proxy that ensures that
7     print statements from other threads won't destroy the prompt. (They
8     will be printed above the prompt instead.)
9     :param return_asyncio_coroutine: When True, return a asyncio coroutine. (Python >3.3)
10    :param true_color: When True, use 24bit colors instead of 256 colors.
11    :param refresh_interval: (number; in seconds) When given, refresh the UI
12    every so many seconds.
13    """
14    assert isinstance(application, Application)
15
16    if return_asyncio_coroutine:
17        eventloop = create_asyncio_eventloop()
18    else:
19        eventloop = eventloop or create_eventloop()
20
21    # Create CommandLineInterface.
22    cli = CommandLineInterface(
23        application=application,
24        eventloop=eventloop,
25        output=create_output(true_color=true_color))
26
27    # Set up refresh interval.
28    if refresh_interval:
29        done = [False]
30        def start_refresh_loop(cli):
31            def run():
32                while not done[0]:
33                    time.sleep(refresh_interval)
34                    cli.request_redraw()
35            t = threading.Thread(target=run)
36            t.daemon = True
37            t.start()
38
39        def stop_refresh_loop(cli):
40            done[0] = True
41
42        cli.on_start += start_refresh_loop
43        cli.on_stop += stop_refresh_loop
44
45    # Replace stdout.
46    patch_context = cli.patch_stdout_context(raw=True) if patch_stdout else DummyContext()
47
48    # Read input and return it.
49    if return_asyncio_coroutine:
50        # Create an asyncio coroutine and call it.
51        exec_context = {'patch_context': patch_context, 'cli': cli,
52                       'Document': Document}
53        exec(textwrap.dedent("""
54    def prompt_coro():
55        # Inline import, because it slows down startup when asyncio is not
56        # needed.
57        import asyncio
58        @asyncio.coroutine
59        def run():
60            with patch_context:
61                result = yield from cli.run_async()
62                if isinstance(result, Document): # Backwards-compatibility.
63                    return result.text
64            return result
65        return run()
66    """), exec_context)
67
68        return exec_context['prompt_coro']()
69    else:
70        try:
71            with patch_context:
72                result = cli.run()

```

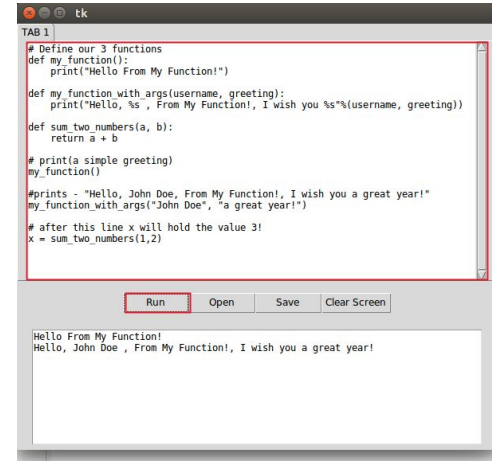
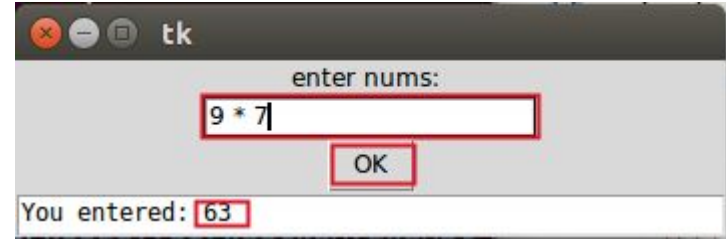
## Plan B

- ❖ Changed strategy and started to build simple python REPL (command-line)
- ❖ Implemented this REPL into simple GUI to understand READ-EVAL-PRINT-LOOP

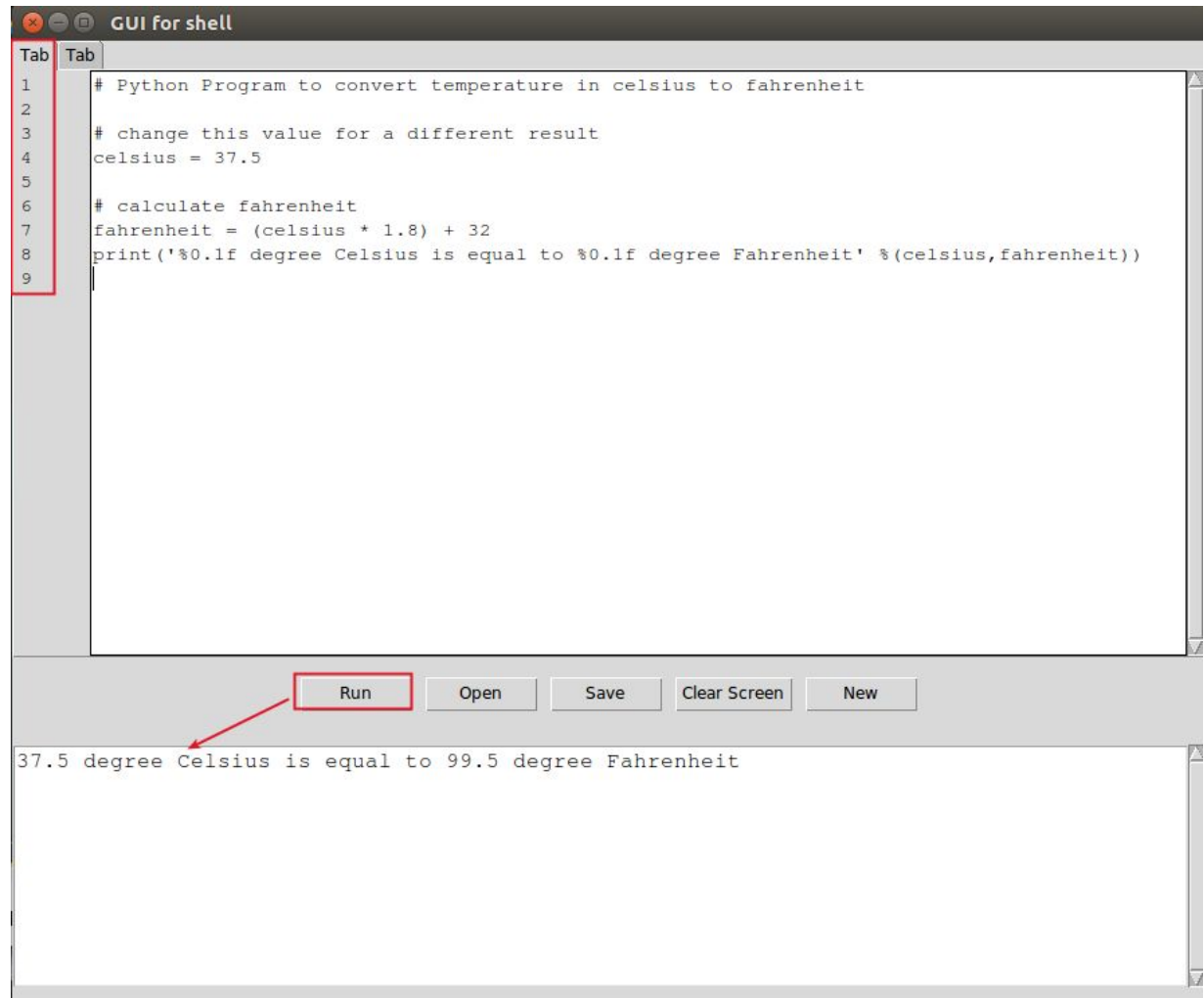


## Basic GUI for python REPL

- ✓ Redirect output from command-line to GUI
- ✓ Redirect input from simple entry box
- ✓ Added textbox and implemented multi-line editing



- ✓ Integrated notebook with scrollbar, line numbers, tabs
- ✓ Multiline editing
- ✓ Open, Save and Run File
- ✓ Clear Screen
- ✓ Open New Tab
- ✓ Handles exceptions





## Review - what is done from last iteration

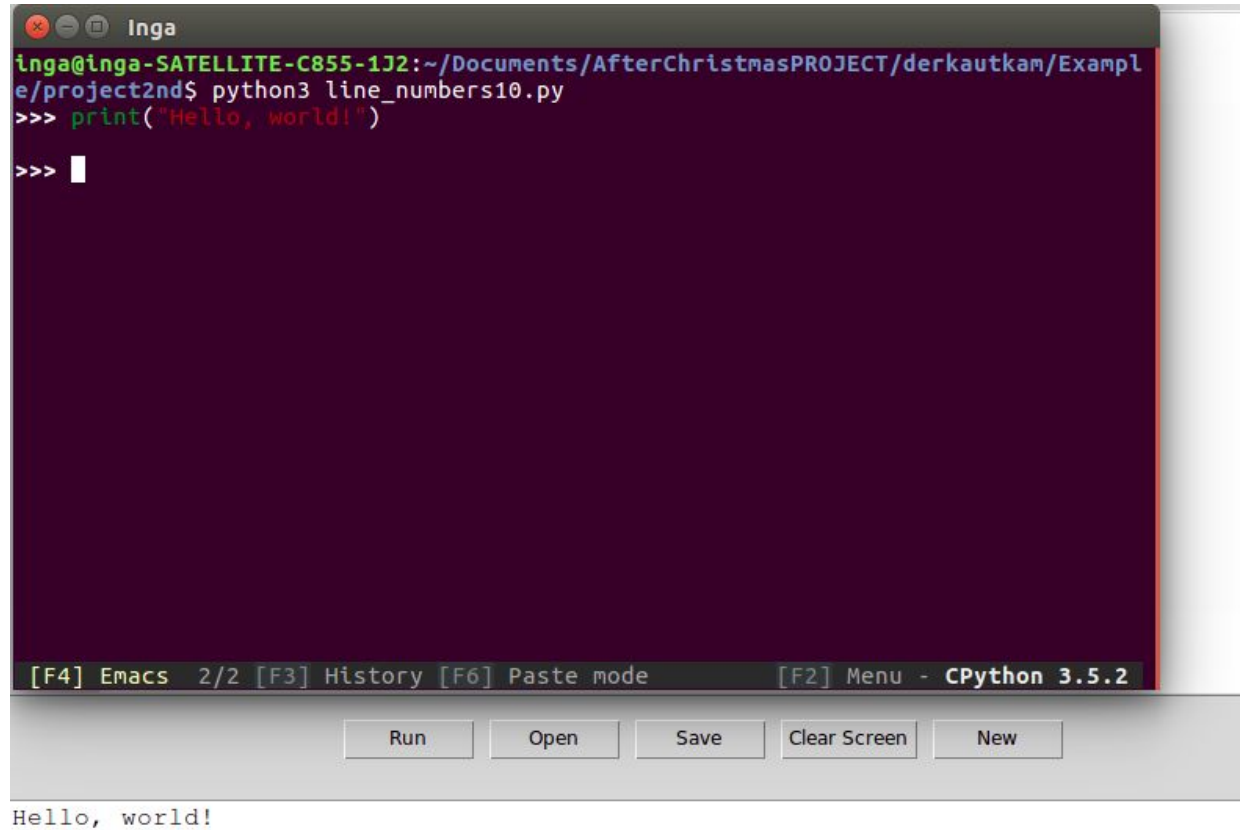
- ✓ Changed strategy and started to build simple GUI for python to understand READ-EVAL-PRINT-LOOP
- ✓ Build basic GUI for python REPL
- ✓ With simple entry box, then added textbox with multiline editing
- ✓ Implemented notebook with scrollbar, line numbers , tabs

# Currently working on

Implementing ppython  
prompt into GUI



- ✓ Embedded Ptpython
- ✓ Output is redirected into gui text box



```
lnge@lnge-SATELLITE-C855-1J2:~/Documents/AfterChristmasPROJECT/derkautkam/Example/project2nd$ python3 line_numbers10.py
>>> print("Hello, world!")
>>>
```

[F4] Emacs 2/2 [F3] History [F6] Paste mode [F2] Menu - CPython 3.5.2

Run Open Save Clear Screen New

Hello, world!

## Plan for next iteration

- ❖ Implement Ptpython into GUI
- ❖ Improve GUI framework
- ❖ Testing



THANK

YOU

THANK

YOU