

```
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

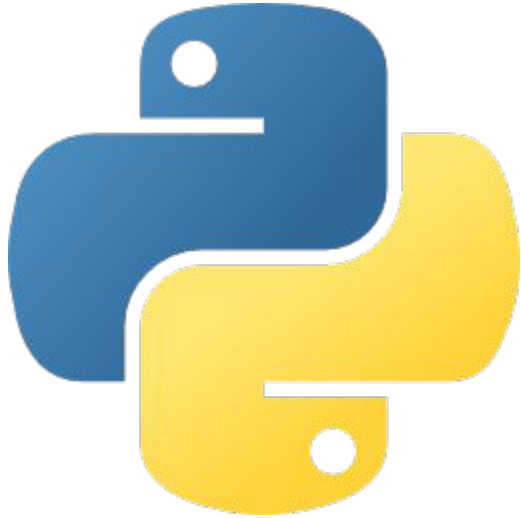
Student Name: **Inga Melkerte**

Student ID: **C00184799**

Supervisor: **Paul Barry**

Date: **15th December 2016**

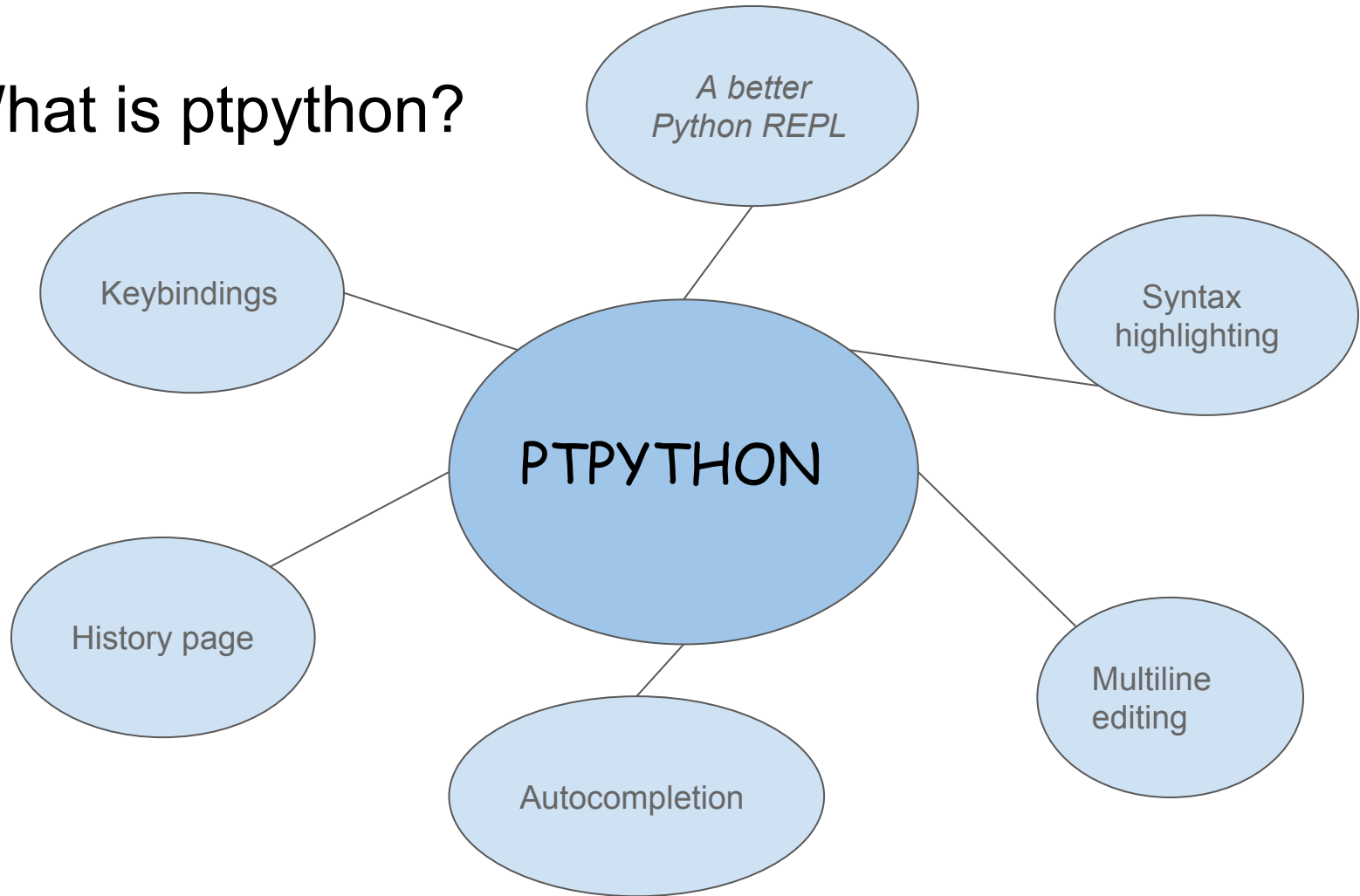
IDLE - Python's Integrated Development and Learning Environment



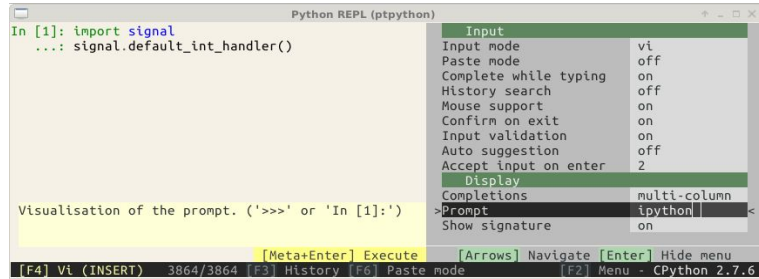
```
Python 3.5.2 Shell
File Edit Shell Debug Options Window Help
Python 3.5.2 (default, Nov 17 2016, 17:05:23)
[GCC 5.4.0 20160609] on linux
Type "copyright", "credits" or "license()" for more
information.
>>> |
```

Ln: 4 Col: 4

What is ptpython?



F2 - Configuration Menu



The screenshot shows the Python REPL configuration menu. The menu is divided into several sections: Input, Display, Completions, and Show signature. The 'Input' section includes options like Input mode, Paste mode, Complete while typing, History search, Mouse support, Confirm on exit, Input validation, Auto suggestion, and Accept input on enter. The 'Display' section includes Display, Completions, and Prompt. The 'Completions' section includes multi-column and ipython. The 'Show signature' section includes on. The status bar at the bottom shows [F4] Vi (INSERT) 3864/3864 [F3] History [F6] Paste mode [F2] Menu - CPython 2.7.6.

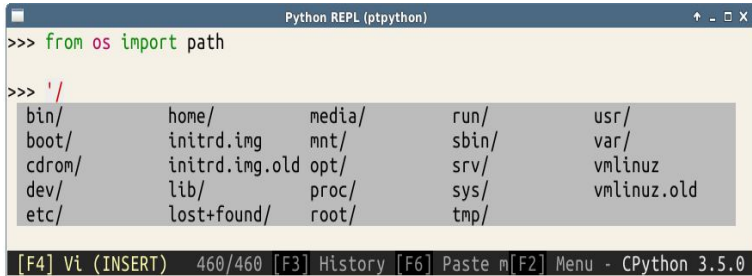
```
Python REPL (ptpython)
In [1]: import signal
...: signal.default_int_handler()

Input
Input mode          vi
Paste mode          off
Complete while typing on
History search      off
Mouse support       on
Confirm on exit     on
Input validation    on
Auto suggestion     off
Accept input on enter 2

Display
Completions         multi-column
Prompt              ipython
Show signature      on

[Meta+Enter] Execute [Arrows] Navigate [Enter] Hide menu
[F4] Vi (INSERT) 3864/3864 [F3] History [F6] Paste mode [F2] Menu - CPython 2.7.6
```

Multiline Editing



The screenshot shows the Python REPL with multiline editing. The prompt is >>> from os import path. The output is a list of directories: bin/, boot/, cdrom/, dev/, etc/, home/, initrd.img, initrd.img.old, lib/, lost+found/, media/, mnt/, opt/, proc/, root/, run/,/sbin/, tmp/, srv/, vmlinuz, vmlinuz.old, and usr/, var/. The status bar at the bottom shows [F4] Vi (INSERT) 460/460 [F3] History [F6] Paste m[F2] Menu - CPython 3.5.0.

```
Python REPL (ptpython)
>>> from os import path

>>> '/'
bin/      home/    media/   run/     usr/
boot/     initrd.img mnt/     sbin/    var/
cdrom/    initrd.img.old opt/     srv/     vmlinuz
dev/      lib/     proc/    sys/     vmlinuz.old
etc/      lost+found/ root/    tmp/

[F4] Vi (INSERT) 460/460 [F3] History [F6] Paste m[F2] Menu - CPython 3.5.0
```

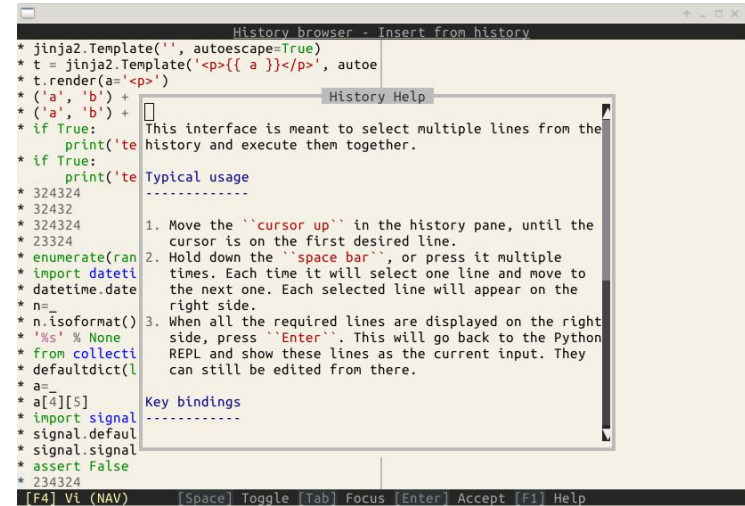
Syntax Validation



The screenshot shows the Python REPL with a syntax error. The prompt is In [1]: print('hello'). The error message is Syntax Error (line=1 column=14). The status bar at the bottom shows [F4] Emacs 298/298 [F6] Paste mode.

```
Terminal
In [1]: print('hello')
Syntax Error (line=1 column=14)
[F4] Emacs 298/298 [F6] Paste mode
```

F3 - History Page and its help



The screenshot shows the Python REPL history page and its help. The history page lists several lines of code, including Jinja2 template rendering and datetime operations. The help page is titled 'History Help' and contains the following text: 'This interface is meant to select multiple lines from the history and execute them together.' The 'Typical usage' section lists three steps: 1. Move the 'cursor up' in the history pane, until the cursor is on the first desired line. 2. Hold down the 'space bar', or press it multiple times. Each time it will select one line and move to the next one. Each selected line will appear on the right side. 3. When all the required lines are displayed on the right side, press 'Enter'. This will go back to the Python REPL and show these lines as the current input. They can still be edited from there. The status bar at the bottom shows [F4] Vi (NAV) [Space] Toggle [Tab] Focus [Enter] Accept [F1] Help.

```
History browser - Insert from history
* jinja2.Template('', autoescape=True)
* t = jinja2.Template('<p>{{ a }}</p>', autoe
* t.render(a='<p>')
* ('a', 'b') +
* ('a', 'b') +
* if True:
  print('te
* if True:
  print('te
* 324324
* 32432
* 324324
* 23324
* enumerate(ran
* import dateti
* datetime.date
* n=
* n.isoformat()
* '%s' % None
* from collecti
* defaultdict(l
* a=
* a[4][5]
* import signal
* signal.default
* signal.signal
* assert False
* 234324
* 234324

History Help
This interface is meant to select multiple lines from the
history and execute them together.

Typical usage
-----
1. Move the "cursor up" in the history pane, until the
cursor is on the first desired line.
2. Hold down the "space bar", or press it multiple
times. Each time it will select one line and move to
the next one. Each selected line will appear on the
right side.
3. When all the required lines are displayed on the right
side, press "Enter". This will go back to the Python
REPL and show these lines as the current input. They
can still be edited from there.

Key bindings
-----

[F4] Vi (NAV) [Space] Toggle [Tab] Focus [Enter] Accept [F1] Help
```

Autocompletion



The screenshot shows the Python REPL with autocompletion. The prompt is >>> if True: print('Hello world') print('你好') os.pa. The autocompletion menu shows the following options: pardir, path, pathconf, pathconf_names, and pathsep. The status bar at the bottom shows [F4] Vi (INSERT) 462/463 [F3] History [F6].

```
Python REPL (ptpython)
>>> if True:
...     print('Hello world')
...     print('你好')
...     os.pa

  pardir
  path
  pathconf
  pathconf_names
  pathsep

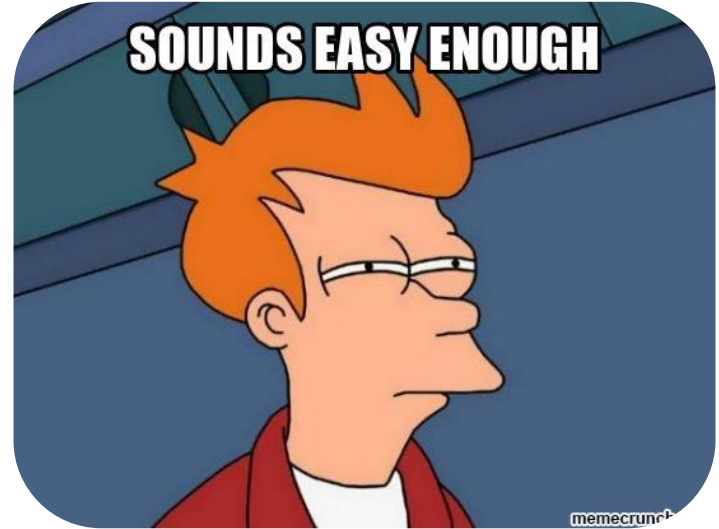
[F4] Vi (INSERT) 462/463 [F3] History [F6]
```

PLAN - build portable GUI for ptpython shell

- Research GUI frameworks
- Get familiar with ptpython
- Build similar GUI like IDLE



SOUNDS
EASY -
DOESN'T?



SO FAR ...

- Looked at ptpython code
- Decided to look at prompt-toolkit
- Difficulty of code
- Chosen GUI framework - Tkinter



yy16384 committed with jonathansenders Fix non-ASCII characters breaking python2		Latest commit 28 days ago
...		
clipboard	Implementation of the Emacs kill-ring (yank-pop command) + unit test...	a month ago
contrib	Remove set comprehension to support python 2.5	4 months ago
eventloop	Handle race condition in eventloop/postix.py	a month ago
filters	Some improvements regarding the multiple cursor support.	2 months ago
key_binding	Fix non-ASCII characters breaking python2	2 days ago
layout	Visualize multiple cursors.	2 months ago
styles	Visualize multiple cursors.	2 months ago
terminal	Fix misup in the mapping from ansi color names to VT100 codes.	a month ago
init.py	Release 1.0.9	a month ago
application.py	A few changes regarding the display of readline-like autocompletions.	6 months ago
auto_suggest.py	Improve documentation.	a year ago
buffer.py	Imple	
buffer_mapping.py	Remo	
cache.py	Add	
completion.py	Bug fi	
document.py	Imple	
enums.py	Renan	
history.py	A few	
input.py	Fixed	
interface.py	Fix 'co	
keys.py	Add	
mouse_events.py	Renan	
output.py	Add	
reactive.py	Retur	
renderer.py	Small	
search_state.py	Use Ti	
selection.py	Imple	
shortcuts.py	Take	
token.py	Suppc	
utils.py	Impro	
validation.py	A few	
win32_types.py	Bug fi	

```
return HSplit([
    VSplit([
        Window(width=D.exact(1), height=D.exact(1),
            content=FillControl(BORDER.TOP_LEFT, token=Token.Window.Border)),
        TokenListToolBar(
            get_tokens=lambda cli: [(Token.Window.Title, ' %s ' % title)],
            align_center=True,
            default_char=Char(BORDER.HORIZONTAL, Token.Window.Border)),
        Window(width=D.exact(1), height=D.exact(1),
            content=FillControl(BORDER.TOP_RIGHT, token=Token.Window.Border)),
    ]),
    VSplit([
        Window(width=D.exact(1),
            content=FillControl(BORDER.VERTICAL, token=Token.Window.Border)),
        body,
        Window(width=D.exact(1),
            content=FillControl(BORDER.VERTICAL, token=Token.Window.Border)),
    ]),
    VSplit([
        Window(width=D.exact(1), height=D.exact(1),
            content=FillControl(BORDER.BOTTOM_LEFT, token=Token.Window.Border)),
        Window(height=D.exact(1),
            content=FillControl(BORDER.HORIZONTAL, token=Token.Window.Border)),
        Window(width=D.exact(1), height=D.exact(1),
            content=FillControl(BORDER.BOTTOM_RIGHT, token=Token.Window.Border)),
    ]),
])
```

1166 lines (949 sloc) | 40.4 KB

```
1 """
2 The main `CommandLineInterface` class and logic.
3 """
4 from __future__ import unicode_literals
5
6 import functools
7 import os
8 import signal
9 import six
10 import sys
11 import time
12 import typing
13 import weakref
```

1166 lines (949 sloc) | 40.4 KB

40.4 KB

```
17 from subprocess import Popen
```

```
19 from .application import Application, ApplicationError
20 from .buffer import Buffer
21 from .buffer_mapping import BufferMapping
22 from .completion import CompleteEvent, get_completion_candidates
23 from .enums import SEARCH_BUFFER
24 from .eventloop.base import EventLoop
```

731 lines (601 sloc) | 27.9 KB

```
25 from .eventloop.interface import EventLoopInterface
26 from .eventloop.loop import EventLoop
27 from .eventloop.loop import EventLoop
28 from .eventloop.loop import EventLoop
29 from .key_binding.input_processor import KeyPress
30 from .key_binding.registry import Registry
31 from .key_binding.vi_state import ViState
```



```
1631 assert isinstance(content, Container)
1632
1633 self.content = content
1634 self.filter = to_cli_filter(filter)
```

```
def __repr__(self):
    return 'ConditionalContainer(%r, filter=%r)' % (self.content, self.filter)
```

```
self):
    content.reset()
```

```
width(self, cli, max_available_width):
    r(cli):
        f.content.preferred_width(cli, max_available_width)
```

```
it(
    , width, max_available_height):
(cli):
```

```
lf.content.preferred_height(cli, width, max_available_height)
```

```
LayoutDimension.exact(0)
```

```
co_screen(self, cli, screen, mouse_handlers, write_position):
```

```
self.filter(cli):
```

```
return self.content.write_to_screen(cli, screen, mouse_handlers, write_position)
```

535 lines (421 sloc) | 18 KB

```
def
1659 return self.content.walk(cli)
```

1660

1661

1662 # Deprecated alias for 'Container'.

1664 lines (1375 sloc) | 66.1 KB