ipynb-creator

A python program that will:

- Read a python file and create a Jupyter ipynb file.
- Read a text file and based on delimiters will create a Jupyter ipynb file with multiple cells.

Recommended prerequisite presentation:
ipynb modification using python json module
... to better understand the json involved.

Ian Stewart
2019-08-07

ipynb-creator

Presentation Contents:

Ian Stewart

2019-08-07

ipynb highlevel layout.

4 x Dictionary keywords

```
{
  "cells": [
    {cell 1 dict}, {cell 2 dict}, {cell ... dict}
    ]
  "metadata": { ~15 overall keyword:value pairs}
  "nbformat": 4,
  "nbformat_minor": 2
}
```

List of dictionaries

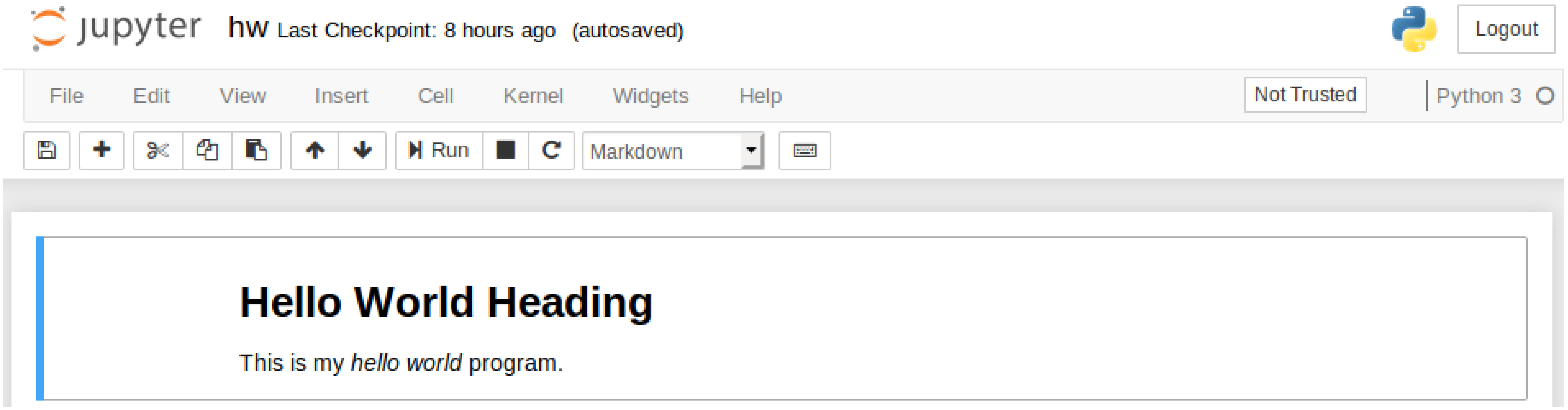
- A dictionary with 4 x keywords: cells, metadata, nbformat and nbformat_minor.
- For “cells” keyword the value is a list.
- The list items are dictionaries for each Jupyter cell.
- **Markdown** cell dictionaries have 3 keywords
- **Code** cell dictionaries have 5 x keywords.

“Markdown” cell ipynb data layout.

```
{
 "cell_type": "markdown",
 "metadata": {},
 "source": [
  "# Hello World Heading\nThis is my *hello world* program.\n"
 ]
},
```

3 x Dictionary keywords:
cell_type, metadata, source.

Jupyter renders this markdown cell as...



Hello World Heading

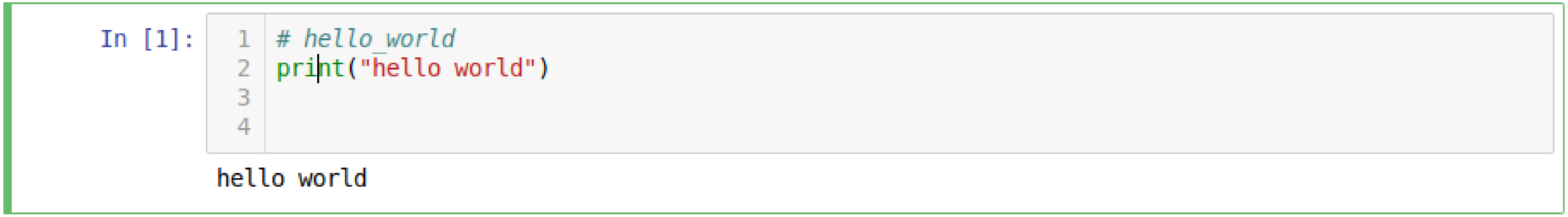
This is my *hello world* program.

"Code" cell ipynb data layout.

```
{
 "cell_type": "code",
 "execution_count": null,
 "metadata": {},
 "outputs": [],
 "source": [
  "# hello_world\nprint(\"hello world\")\n\n"
 ]
},
```

5 x Dictionary keywords: cell_type, execution_count, metadata, outputs, source.

Jupyter renders this code cell as...



```
In [1]:  # hello_world
         print("hello world")


hello world
```

Logic to the python program

- Create a simple template as a python constant. The template has 1 x markdown cell.
- Select a python file or text file to become the ipynb file.
- Having determined the ipynb file name write the template out to the ipynb file.
- Import simplejson.
- Use json to load ipynb file and change markdown text.
- Use json.dump() to write json data back to the file.
- Use json.load(), add cell, and json.dump().
- As required, repeat above step until all cells have been written.

Create ipynb template 1/2

```
TEMPLATE = """{
 "cells": [
  {
   "cell_type": "markdown",
   "metadata": {},
   "source": [
    "template"
   ]
  }
 ],
 "metadata": {
  "kernelspec": {
   "display_name": "Python 3",
   "language": "python",
   "name": "python3"
  },
```

1st top level dictionary: “cell_type”

Dummy markdown data

2nd top level dictionary: “metadata”

Create ipynb template 2/2

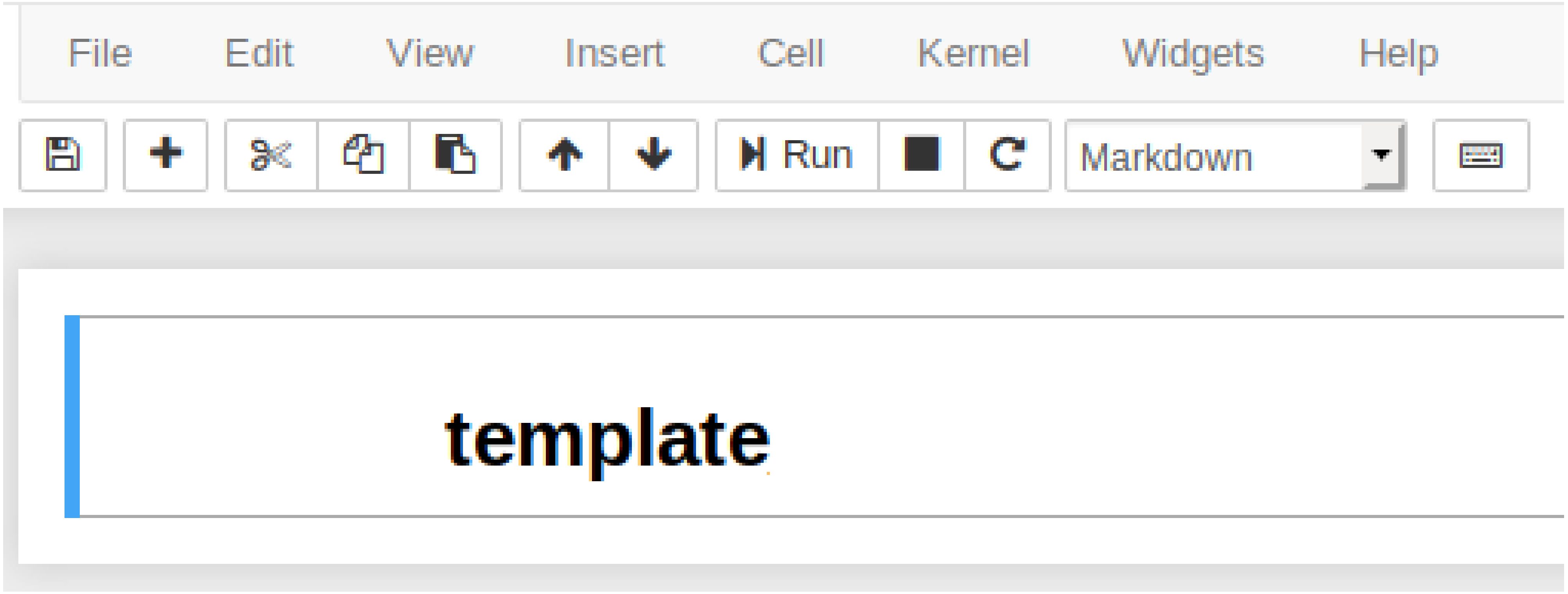
```
  "language_info": {
   "codemirror_mode": {
    "name": "ipython",
    "version": 3
   },
   "file_extension": ".py",
   "mimetype": "text/x-python",
   "name": "python",
   "nbconvert_exporter": "python",
   "pygments_lexer": "ipython3",
   "version": "3.6.8"
  }
 },
 "nbformat": 4,
 "nbformat_minor": 2
}"""
```

2nd top level dictionary continues

3rd & 4th top level dictionaries

ipynb-creator program logic

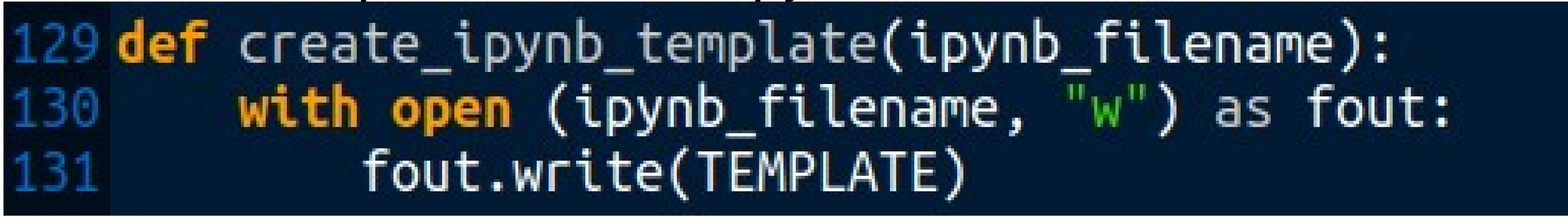
If the TEMPLATE was rendered by Jupyter notebook...



template

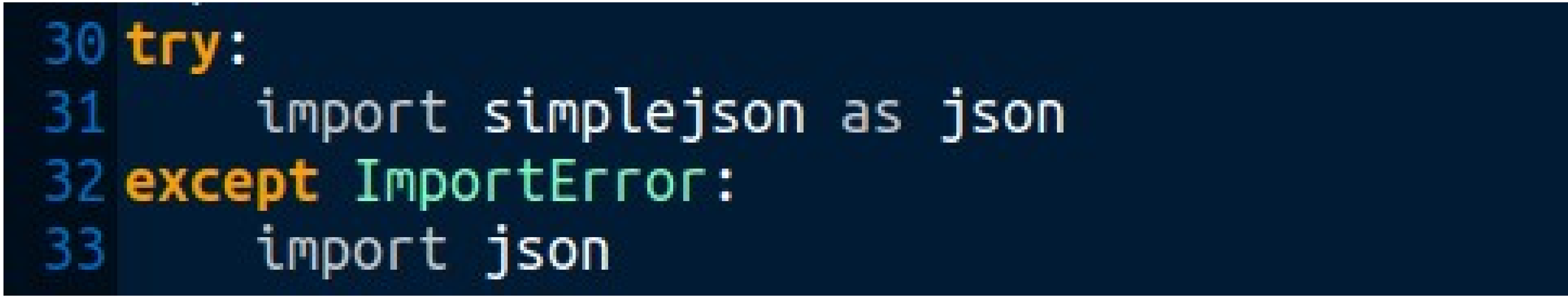
ipynb-creator program logic

Write the template out to an ipynb file



```
def create_ipynb_template(ipynb_filename):
    with open (ipynb_filename, "w") as fout:
        fout.write(TEMPLATE)
```

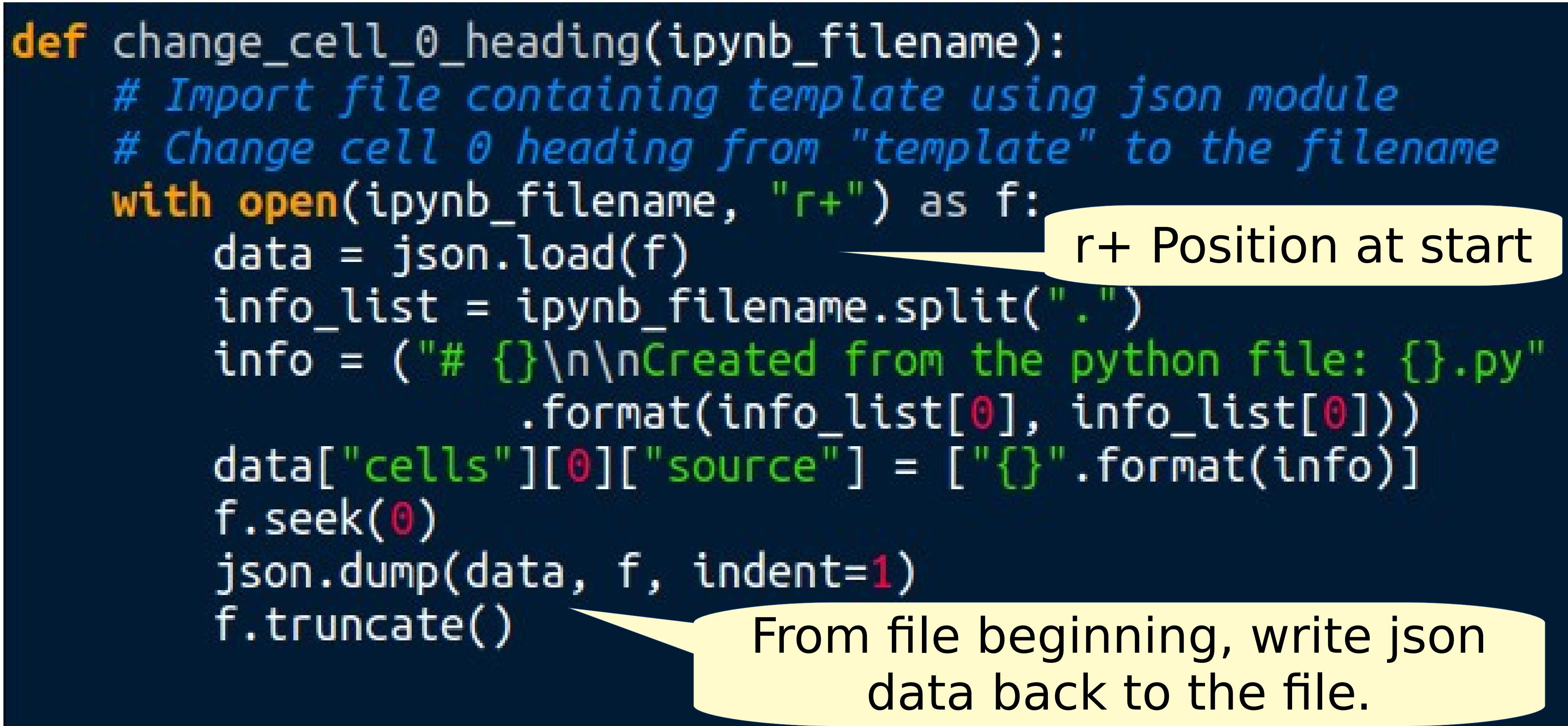
Import simplejson module



```
try:
    import simplejson as json
except ImportError:
    import json
```

ipynb-creator program logic

Use json to load ipynb file and change markdown text



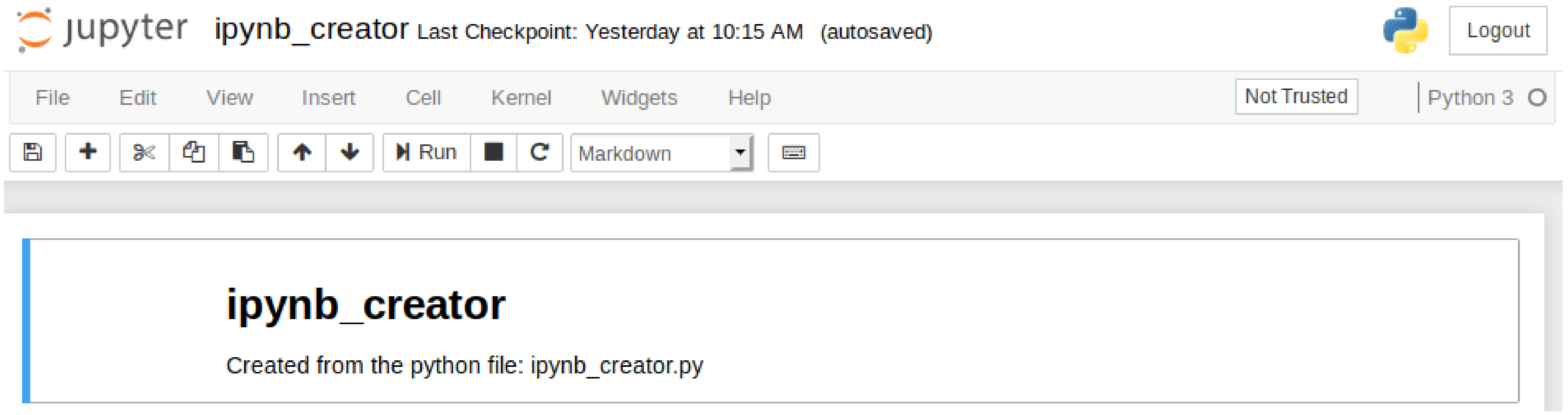
```python
def change_cell_0_heading(ipynb_filename):
    # Import file containing template using json module
    # Change cell 0 heading from "template" to the filename
    with open(ipynb_filename, "r+") as f:
        data = json.load(f)
        info_list = ipynb_filename.split(".")
        info = ("# {}\n\nCreated from the python file: {}.py"
                    .format(info_list[0], info_list[0]))
        data["cells"][0]["source"] = ["{}".format(info)]
        f.seek(0)
        json.dump(data, f, indent=1)
        f.truncate()
```

r+ Position at start

From file beginning, write json data back to the file.

ipynb-creator program logic

If the ipynb file is rendered by Jupyter notebook...

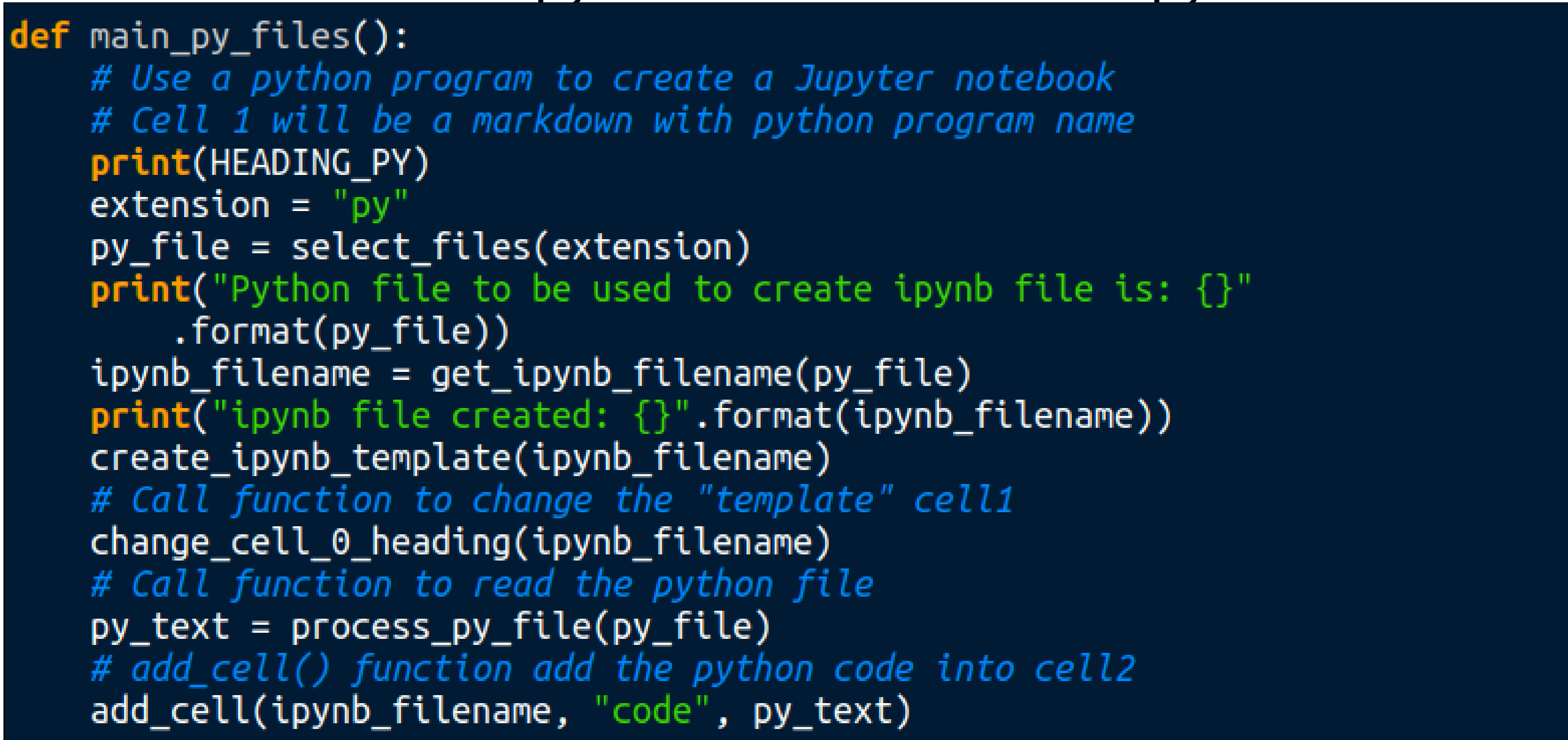


ipynb_creator

Created from the python file: ipynb_creator.py

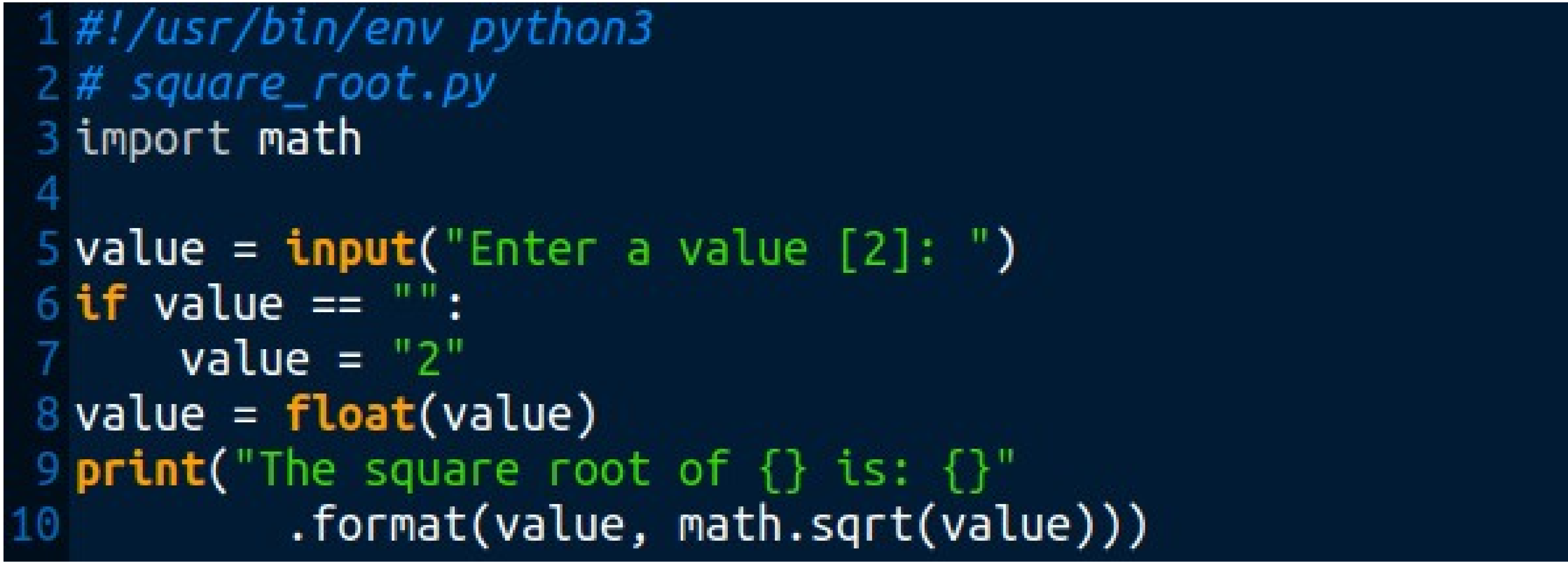
ipynb-creator program logic

Functions to read a python file and add it to ipynb cell 2...



```python
def main_py_files():
    # Use a python program to create a Jupyter notebook
    # Cell 1 will be a markdown with python program name
    print(HEADING_PY)
    extension = "py"
    py_file = select_files(extension)
    print("Python file to be used to create ipynb file is: {}"
        .format(py_file))
    ipynb_filename = get_ipynb_filename(py_file)
    print("ipynb file created: {}".format(ipynb_filename))
    create_ipynb_template(ipynb_filename)
    # Call function to change the "template" cell1
    change_cell_0_heading(ipynb_filename)
    # Call function to read the python file
    py_text = process_py_file(py_file)
    # add_cell() function add the python code into cell2
    add_cell(ipynb_filename, "code", py_text)
```

Python square_root.py program example 1/3



```python
#!/usr/bin/env python3
# square_root.py
import math

value = input("Enter a value [2]: ")
if value == "":
    value = "2"
value = float(value)
print("The square root of {} is: {}"
        .format(value, math.sqrt(value)))
```

```
$ python3 square_root.py
Enter a value [2]: 3
The square root of 3.0 is: 1.7320508075688772
```

Python square_root.py program example 2/3

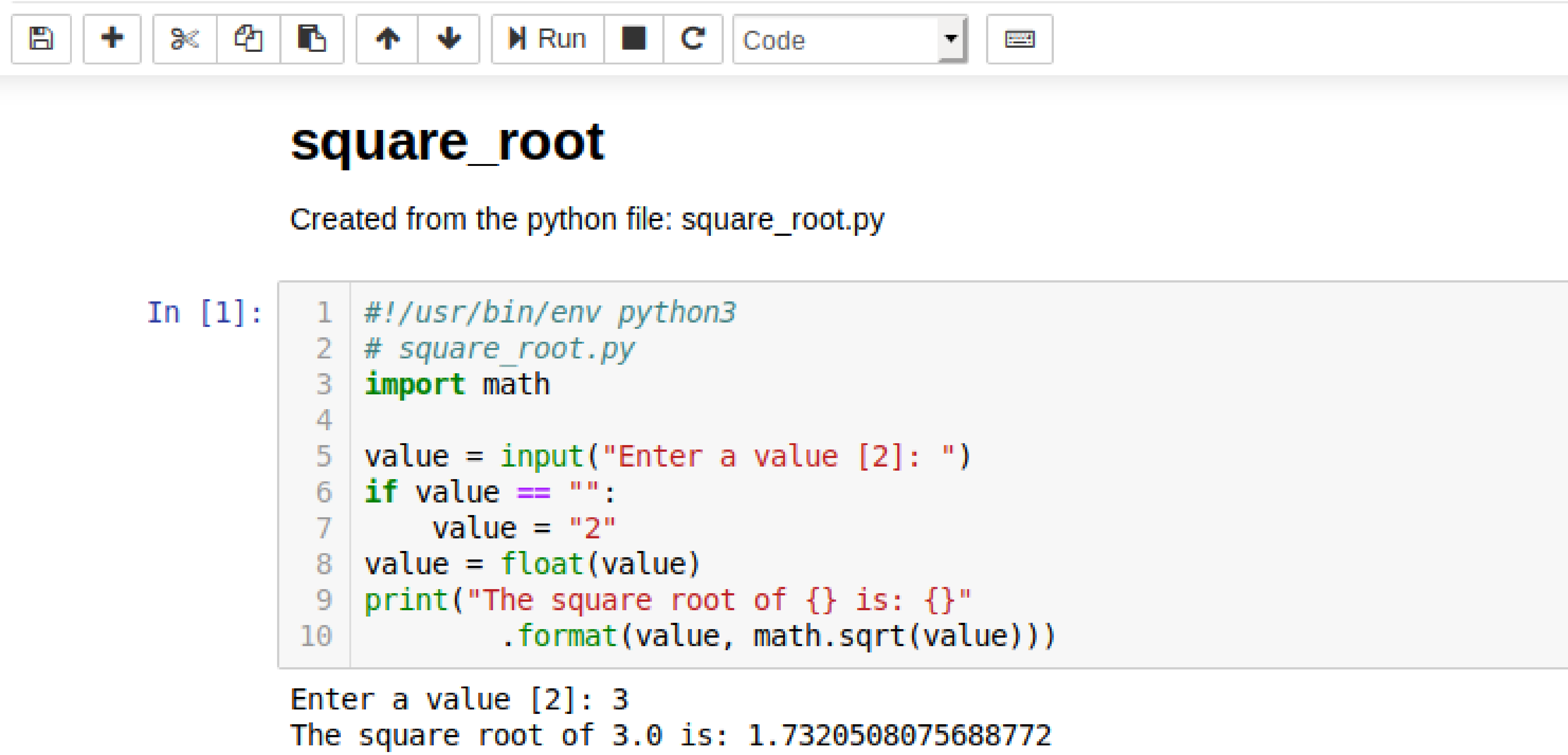
```
$ python3 ipynb-creator.py
ipynb-creator version: 0.3

Move the contents of a python file to Jupyter notebook?
[Y/n]:

Read a python file and create an ipynb file.
Python files (.py) found in the current directory:
  1. ipynb-creator.py
  2. square_root.py

Select the file for creating the ipynb file [1]: 2
Python file to be used to create ipynb file is:
square_root.py
ipynb file created: square_root.ipynb
```

Python square_root.py program example 3/3



square_root

Created from the python file: square_root.py

In [1]:

```python
#!/usr/bin/env python3
# square_root.py
import math

value = input("Enter a value [2]: ")
if value == "":
    value = "2"
value = float(value)
print("The square root of {} is: {}"
        .format(value, math.sqrt(value)))
```

```
Enter a value [2]: 3
The square root of 3.0 is: 1.7320508075688772
```

ipynb-creator -h and --help

```
$ python3 ipynb-creator.py -h

ipynb-creator version: 0.3
Usage: ipynb-creator [OPTION]... [FILE]...
Create Jupyter notebook ipynb file(s) upon having been supplied python
(.py) or text (.txt) file(s)

[OPTION]...
Options and arguments:
   -h       print this brief help message and exit.
   --help   print the full help message which includes an example then exit.

[FILE]...
If no files are provided as arguments then the program will run in a menu
driven mode.
```

ipynb-creator --help text example 1/3

```
Anything written here is ignored because its before the first delimiter.
This file is stored in my github repository and in my /python/dev/ folder.
I wrote this text in August 2019.

<markdown>
# Hello World Heading
This is my *hello world* program.
<code>
# hello_world
print("hello world")

< markdown The second python program will do some maths.>
# Maths
This is how to obtain the **square root of 2**
```

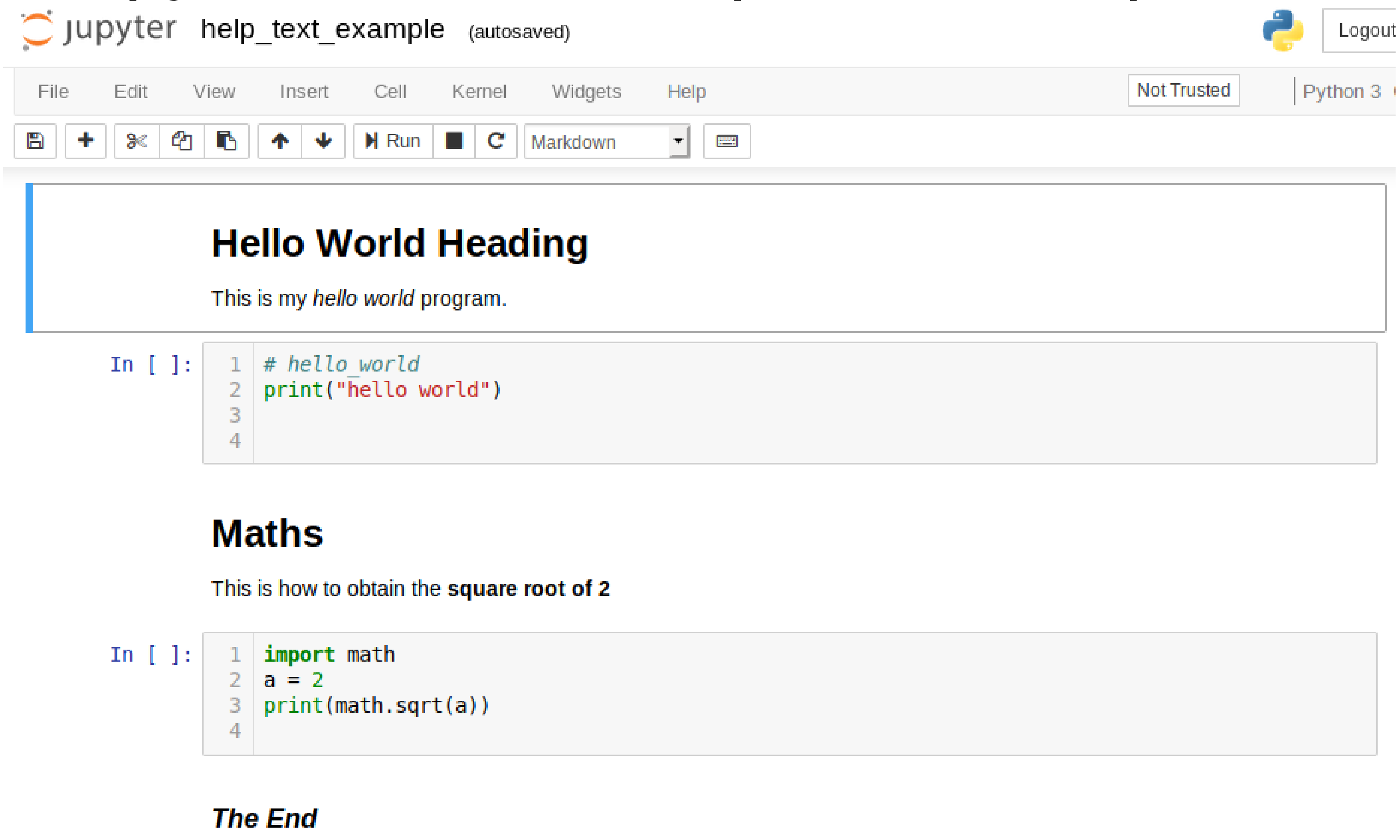
ipynb-creator --help text example 2/3

```
< code >
import math
< comment Remember to include the import math!>
a = 2
print(math.sqrt(a))
< markdown >
### *The End*
<comment This is the end of the hello_world.txt example file.>
```

```
$ python3 ipynb-creator.py help_text_example.txt

ipynb-creator version: 0.3
ipynb file created: help_text_example.ipynb
Total cells in ipynb file: 5
```

ipynb-creator --help text example 3/3


jupyter help_text_example (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Not Trusted
Python 3
Run
Markdown

Hello World Heading

This is my *hello world* program.

```
In [ ]:
# hello_world
print("hello world")
```

Maths

This is how to obtain the **square root of 2**

```
In [ ]:
import math
a = 2
print(math.sqrt(a))
```

The End

ipynb-creator --help text delimiter rules

For the text (.txt) files the delimiter guidelines are:

o Delimiters start with left angle bracket "<" and end with right angle ">".
o A delimiters left angle bracket "<" must be the first character on a line.
o Delimiters that create Jupyter notebook cells are <markdown> and <code>.
o Delimiter <raw> is accepted but not processed.
o Delimiter <comment> allows one line comments within the text file.
E.g. < comment The next code cell is from my hello_world.py program>
o Other delimiters may include a comment.
E.g. <code This is my /python/hello_world.py program>
o A delimiter may be surrounded by spaces. E.g. <   code   >
o Text that follows a delimiter becomes the markdown or the code.
o Lines of text before the first delimiter are ignored.

ipynb-creator – launching arguments

Examples of the arguments that may be passed to ipynb-creator on launching

```
$ python3 ipynb-creator.py
$ python3 ipynb-creator.py read_file.py
$ python3 ipynb-creator.py hw.txt hello_world_1.txt sysarg.py
$ python3 ipynb-creator.py *.py
$ python3 ipynb-creator.py *.txt
$ python3 ipynb-creator.py *.txt *.py
```

If ipynb-creator moved to /local/bin/ then...

```
$ ipynb-creator
$ ipynb-creator read_file.py
$ ipynb-creator hw.txt hello_world_1.txt sysarg.py
$ ipynb-creator *.txt *.py
```

ipynb_creator

Questions?
Demos?