

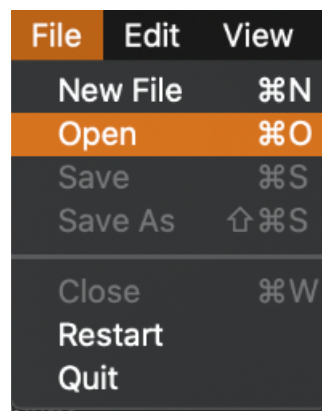
Human Testing Protocol for Issue #1251 - Discarding non-added changes accidentally

1. Launch Porcupine in terminal using “python3 -m porcupine”

```
% python3 -m porcupine
```

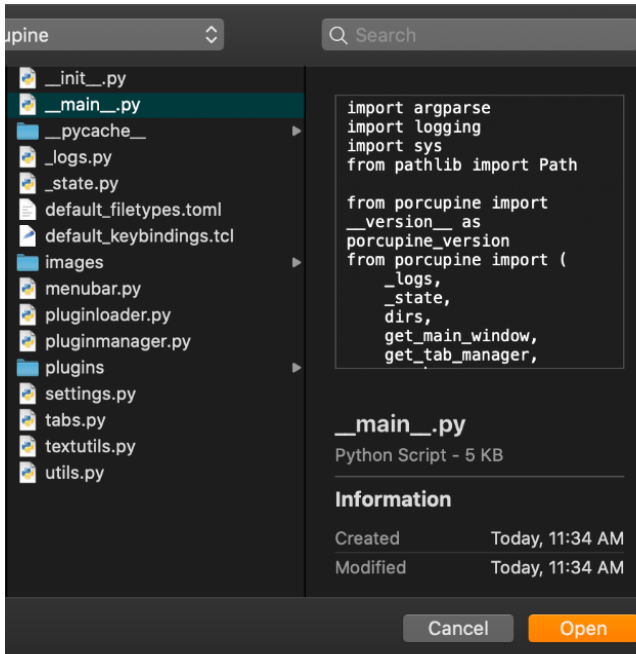
Launching Porcupine in terminal

2. When the application launches, go to “File” on the top bar, then select “Open”.

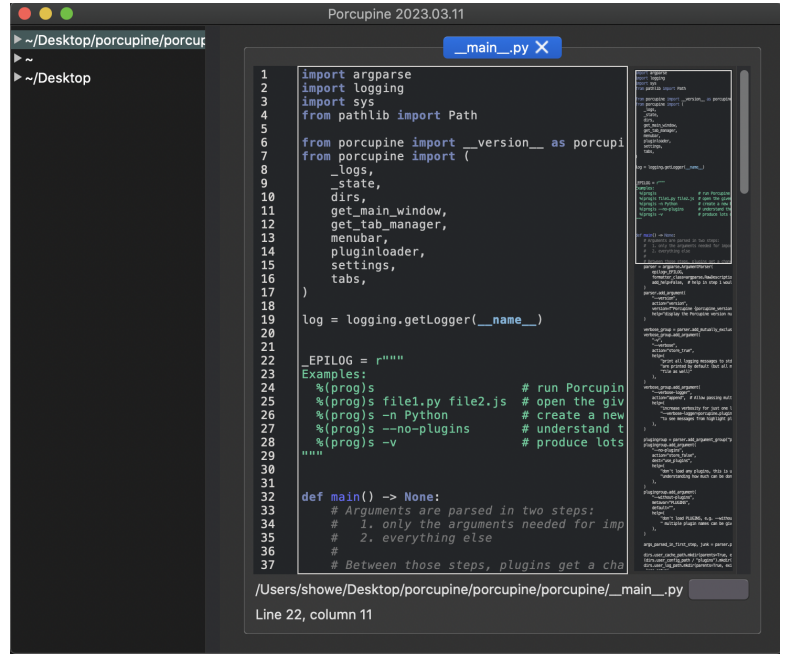


Selecting “Open” in the “File” Menu

3. Now find any type of text file on your machine, this could be a .txt file, .py file, .js, etc., select it and select “Open”. Now you should see it inside the Porcupine Text Editor.

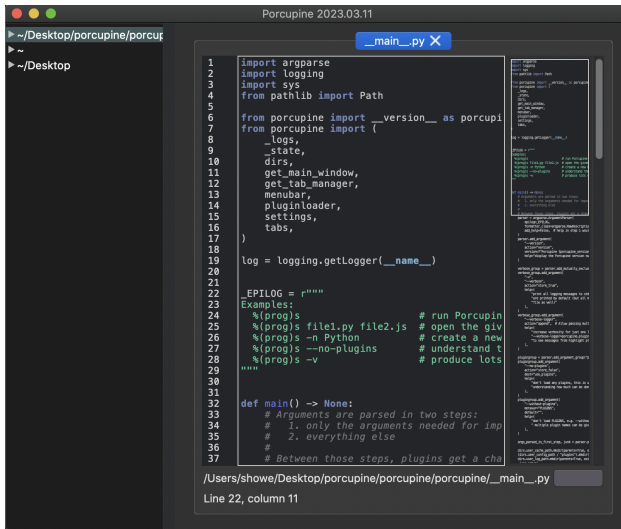


(Left): Selecting `__main__.py` and clicking Open



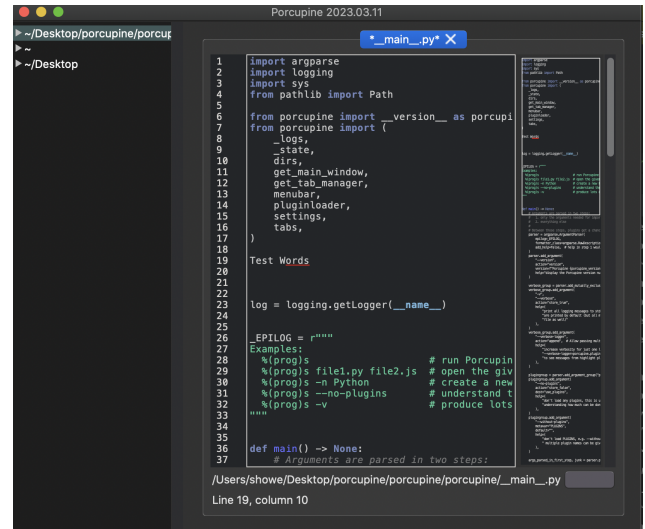
(Right): The results inside the application

- Once it opens, go to any location in the file and add any changes you desire, this could be random text, a comment, a function, a variable, etc.



The screenshot shows the Porcupine IDE with the file `__main__.py` open. The code includes imports for `argparse`, `logging`, `sys`, and `Path`, followed by a `from porcupine import` statement. A `log` object is created using `logging.getLogger(__name__)`. An `EPILOG` string is defined with examples of command-line usage. The `main` function is defined and takes no arguments. The cursor is positioned at line 22, column 11.

```
1 import argparse
2 import logging
3 import sys
4 from pathlib import Path
5
6 from porcupine import __version__ as porcupine_version
7 from porcupine import (
8     _logs,
9     _state,
10     dirs,
11     get_main_window,
12     get_tab_manager,
13     menubar,
14     pluginloader,
15     settings,
16     tabs,
17 )
18
19 log = logging.getLogger(__name__)
20
21
22 EPILOG = """
23 Examples:
24 %(prog)s file1.py file2.js # run Porcupine
25 %(prog)s file1.py file2.js # open the giv
26 %(prog)s -n Python # create a new
27 %(prog)s --no-plugins # understand t
28 %(prog)s -v # produce lots
29 """
30
31 def main() -> None:
32     # Arguments are parsed in two steps:
33     # 1. only the arguments needed for imp
34     # 2. everything else
35
36     # Between those steps, plugins get a cha
```



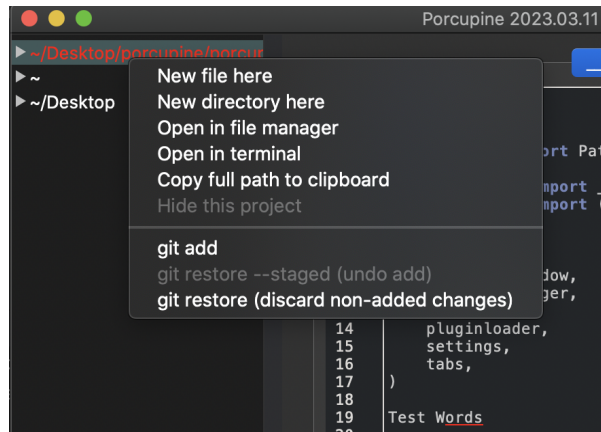
The screenshot shows the Porcupine IDE with the file `__main__.py` open. The code is identical to the previous screenshot, but with the addition of `Test Words` on line 19. The cursor is positioned at line 19, column 10.

```
1 import argparse
2 import logging
3 import sys
4 from pathlib import Path
5
6 from porcupine import __version__ as porcupine_version
7 from porcupine import (
8     _logs,
9     _state,
10     dirs,
11     get_main_window,
12     get_tab_manager,
13     menubar,
14     pluginloader,
15     settings,
16     tabs,
17 )
18
19 Test Words
20
21 log = logging.getLogger(__name__)
22
23
24 EPILOG = """
25 Examples:
26 %(prog)s file1.py file2.js # run Porcupine
27 %(prog)s file1.py file2.js # open the giv
28 %(prog)s -n Python # create a new
29 %(prog)s --no-plugins # understand t
30 %(prog)s -v # produce lots
31 """
32
33 def main() -> None:
34     # Arguments are parsed in two steps:
35     # 1. only the arguments needed for imp
36     # 2. everything else
37
38     # Between those steps, plugins get a cha
```

Before (Left) and After (Right) adding "Test Words" on line 19

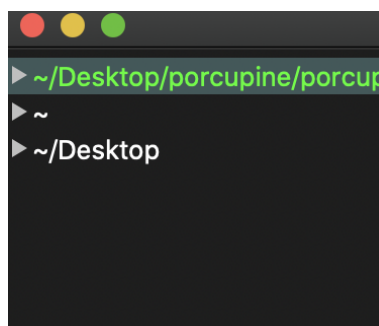
- If you are on a windows or linux machine, press `control+s` to save the changes, if you are on a mac press `command+s`. You'll know its worked if the "*" before and after the filename disappear.

6. Now, on the right hand side you should see the directory where this file is located, and it should be highlighted red, right click it and a git menu should appear.



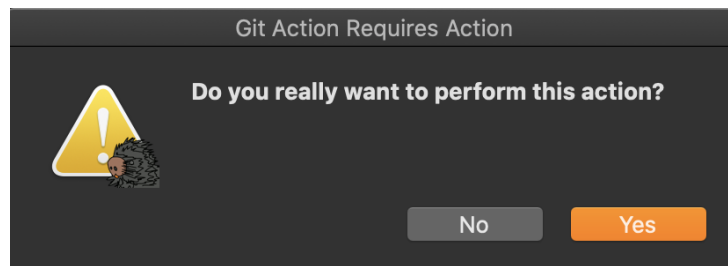
Right Clicking on ~/Desktop/porcupine...

7. Now there are 3 functions we are going to test. They have to be done in order:
- When you click git add, it should not trigger a confirmation pop-up, and the color of the items in the directory that were red should now be green



The Directory is now Green where it was just red

- b. Now, click git restore –staged (undo add), it should also not trigger a confirmation pop-up and the directory should return to the red color, now at this point any changes you’ve made should still be visible in the file.
- c. Now, click git restore (discard non-added changes), this should trigger a confirmation pop-up.



This is the pop-up that should appear

First click no, the pop up should disappear and you should still be able to see any modifications you made to the file. In this file we added “Test Words” on line 19. Now click git restore (discard non-added changes) and click yes on the pop-up, now any changes you made to the file before you saved the file should disappear and the directory should go back to its default color.

If all three of these test perform as described above, then the test pass, otherwise the test fails.