Atom: A Hackable Editor for the 21st Century



Jerry Feldman

BLU 2/19/2020



Overview

- First I will display some information from the Atom Flight Manual.
- Next I will display some screen shots with some of the screen elements that are hard to see in the demo
- Third I will demo a couple of Python scripts I have been working on to show many of the useful features.



The Atom Welcome Screen

This is what you will see when you first run Atom. You can bring up the welcome screen at any time

🔴 🔴 🕘 🔄 🔄 Welcome —	- ~/My Project			
Welcome	Welcome Guide			
	Get to know Atom!			
	Open a Project			
& ATOM	Version control with Git and GitHub			
A hackable text editor for the 21 st Century	^(a) Collaborate in real time with Teletype			
For help, please visit	Install a Package			
 The Atom docs for Guides and the API reference. The Atom forum at discuss.atom.io The Atom org. This is where all GitHub-created Atom packages can be found. 	☺ Choose a Theme			
Show Welcome Guide when opening Atom	Customize the Styling			
atom.io × ₩	↔ Hack on the Init Script			
	↔ Add a Snippet			
	Learn Keyboard Shortcuts			
Welcome	🗘 GitHub 🔶 Git (



Atom: A Hackable Editor for the 21st Century

The Atom Flight Manual

 The Atom Flight Manual is an extensive guide in how to use atom. This will tell you all you need to know about atom:

https://flight-manual.atom.io/



How to get Atom

- Atom can be downloaded directly from: https://flight-manual.atom.io/getting-started/s ections/installing-atom/
- The flight manual has detailed instructions that can be copied and pasted.



Atom: A Hackable Editor for the 21st Century

Why I Chose Atom

- As I mentioned in the abstract, I am a long time EMACS user going back over 30 years. I used EMACS mainly for C, C++, and more recently TCL and Python.
- When I moved to Red Hat, I found there were many things that EMACS failed to do with my Python code, such as trim trailing spaces and syntax check.
- Many on my team used Eclipse which I hate.



Why I Chose Atom

- I looked at Geany, but IMHO it was somewhat light weight for me. One of the main features I used in EMACS was to be able to display files in different windows, both horizontally and vertically.
- Pycharm was a better IDE, but to my knowledge it did not do panes.
- Another developer suggested Atom. Atom pretty much did what I wanted.



The Features I can't live without

- As I mentioned: Panes (In Atom terminology).
- A built in file manager. Atom, Geany and Pycharm have tree views.
- Different themes.
- Note that text editors can be very personal.



Some Screen Shots

rightClick.py — treeview — ~/python/qttest — Atom ×						
File Edit View Selection Find Pack	ages Help					
Project	ded × de	d_ × fileV_ o colu_ × 4lev × righ × sim ×		🗙 emit 🗙 3pin 🗙 ded 🗙 sort 🗙		
 dedup [master] i=pycache git dedup_previous duptest foostuff ions iteststuff gitignore #dedup_excludes.listbox.glade# argv.py columnenums.py dedup_config_qt.py dedup_config.qt.py dedup_excludes.listbox.glade dedup_excludes.glade dedup_excludes.py dedup_excludes.py dedup_main_qt.py dedup_at_rc.py 	471 472 473 474 475 476 477 478 479 480 481 482 483 484 485 486 487 488 489 490 491	<pre>self.logger.setLevel(logging.DE # set text buffer scroll to deb self.handlers = {} #store handl # for logging to file self.logFileName = Path.joinp # for existing users remove ded try: self.logFileName.unlink() except: pass # In logging to file, get the l # this should have a '{}' for t #tofile = self.get_obj("logtofi #logtolabel = tofile.get_label(#tofile.set_label(logtolabel.fo # alter the title to include th title = self.menuLogToHome.titl self.menuLogToHome.setTitle(tit def configureLogging(self, loggingD self.logger.get.get.get.get.get.get.get.get.get.get</pre>	22 c 23 s 24 r 25 p 26 f 27 28 29 30 31 self. 32 self. 33 34 34 #self 35 self. 36 37 38 self. 39 40 41 def click	<pre>ol2 = QtGui.QStandardItem(' ol3 = QtGui.QStandardItem(' elf.uiItems.model().appendR owToAdd = self.uiItems.model arent = self.uiItems.model(or j in range(3): col1 = QtGui.QStandardIt col2 = QtGui.QStandardIt col3 = QtGui.QStandardIt parent.appendRow([col1,c uiItems.expandAll() uiItems.clicked.connect(se uiItems.clicked.connect(se uiItems.customContextMenuRe myMenu = QtWidgets.QMenu('M myMenu.addAction(QtWidgets.) edItem(self, index):</pre>		
 dedup_qt.qrc dedup_qt.ui 				s: Case Insensitive 📪 🛄 🗙		
<pre>dedup_qt.ui dedup_qtfordisplay.py</pre>	compare					
	default					
treeview/rightClick py ① 0 ▲ 0 ① 0 1:1			 LF UTF-8 Python git 	+ 🖗 master 🖸 Fetch 🎧 GitHub 🗢 Git (0)		

File Name errors Warpings and info Line and Column number



Some Screen Shots

× rightClick.py — treeview — ~/python/qttest — Atom File Edit View Selection Find Packages Help fileV.... o colu... × 4lev... × fileV....o ded... × riah... × emit... × 3pin... × ded... × sort... × Project ded × ✓ ☐ dedup [master] col2 = OtGui.OStandardItem(' self.logger.setLevel(logging.DE col3 = QtGui.QStandardItem(' > 🛅 __pycache__ # set text buffer scroll to deb self.uiItems.model().appendR self.handlers = {} #store handl rowToAdd = self.uiItems.mode # for logging to file parent = self.uiItems.model(self.logFileName = Path.joinp for j in range(3): # for existing users remove ded col1 = QtGui.QStandardIt > 🛅 icons col2 = OtGui.OStandardIt self.logFileName.unlink() col3 = QtGui.QStandardIt except: parent.appendRow([col1,c pass self.uiItems.expandAll() self.uiItems.setSortingEnabled(T # this should have a '{}' for t #tofile = self.get obj("logtofi dedup_config_qt.py self.uiItems.customContextMenuRe dedup_config.py # alter the title to include th dedup_excludes.glade self.myMenu = QtWidgets.QMenu('M title = self.menuLogToHome.titl self.myMenu.addAction(QtWidgets. self.menuLogToHome.setTitle(tit dedup_excludes.py def configureLogging(self, loggingD dedup_main_qt.py def clickedItem(self, index): self.logger.debug("configurelog dedup_qt_rc.pv : Case Insensitive 📪 🛄 dedup_at.arc 🖹 dedup_qt.ui compare default 🖹 dedup.glade ● LF UTF-8 Python git+ 🖗 master 🤁 Fetch 🎧 GitHub 👁 Git (0) treeview/rightClick.py ① 0 🛕 0 ③ 0 1:1

LF: Use LF (LineFeed) as end of line. UTF-8 character set Python language Use the git+ package...



Atom: A Hackable Editor for the 21st Century

Some Screen Shots

4 Panes Visible Each pane has its own set of tabs.

Treeview Top Part of code Further on in the same code yet another program Find and Replace

dedup_main_qt.py — dedup — ~/python/dedup — Atom							
File Edit View Selection Find Package	es Help						
Project	aed X ded X	fileV_O cou_x tiev_x tight to sint x ded_ x	fileV o d	led x rightClick.py x emit x 3pin x ded x sort x sort x			
 dedup [master] wrache git dedup_previous duptest foostuff itopsc tteststuff gitignore ttedup_excludes_listbox.glade# tdedup_excludes_listbox.glade# 	approximation 65 definit(self, parent=None): git 66 super(Dedup, self)init(parent=parent) dedup_previous 67 # Get the full path name to the executable depst 68 self.iam = Path(file_).resolve() self.setupUi(self) 69 self.setupUi(self) icore 70 # get_pointer to self namespace. We need this resturf 72 self.NameSpace = vars(self) gitspore 73 if self.actionShowLogsOnStartup.isChecked(): gitspore 73 if self.actionDEBUGConsole.setChecked(True) self.logger.setLevel(logging.DEBUG) # set default logging level dedup_config.yt 470 # set text buffer scroll to debug and console get_dedup_config.yt 471 self.logFileName = Path.joinpath(Path.home(dedup_config.yt 472 # set text buffer scroll to debug and console get_dedup_config.yt 473 self.logFileName = Path.joinpath(Path.home(dedup_config.yt 475 self.logFileName = Path.joinpath(Path.home(dedup_config.yt 476 # for cwisting users remove dedup.log dedup_cotions_state 476 gelf.logFileName.unlink()	21 22 23 24 25 26 27 28 29 30 31 32	<pre>col2 = QtGui.QStandardItem('Item' + str(i) col3 = QtGui.QStandardItem('Item' + str(i) self.uiItems.model().appendRow([col1,col2, rowToAdd = self.uiItems.model().rowCount() parent = self.uiItems.model().item(rowToAd for j in range(3):</pre>				
dedup_config.py dedup_excludes.glade dedup_excludes.glade dedup_excludes.glade= dedup_excludes.listbox.glade=		32 33 34 35 36 37 38 39 40	<pre>self.uiltems.setSortingEnabled(True) #self.uiltems.clicked.connect(self.clickedItem self.uiItems.customContextMenuRequested.connec self.myMenu = QtWidgets.QMenu('Menu', self) self.myMenu.addAction(QtWidgets.QAction('Test'))</pre>				
			<pre>def clickeditem(self, index): print("Clicked") if index.isValid(): rect = self.uiItems.visualRect(index) noc = celf.uiItems.visualRect(index) Finding with Options: Case Insensitive .* Aa : 100 X roc results Find Find All </pre>				
 in dedup.prev.py in dedup.ov dedup.main_qt.py do 0 0 476:26 	default			Replace Replace All ● LF UTF-8 Python git+ IP master C Fetch C Git(0)			



Demo

- I'm going to demo some of Atom's features. The code I will demo is exifdata, a project I created as a graphical forensic tool to view the meta data of an image.
- I will also explain each of the menus that are hard to capture in a screen shot.

