

Task Editor And Launcher (TEAL)

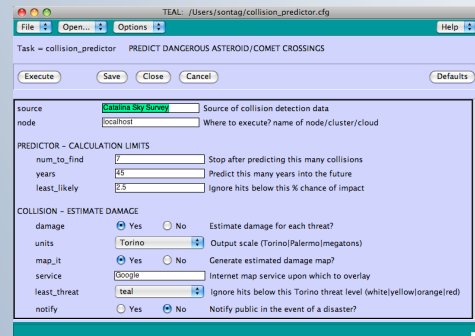


Why write a configuration GUI when you don't have to?

C. Sontag & W. Hack, Space Telescope Science Institute

TEAL ...

- Handles all the parameters for your task/script/program
- Allows your users to safely edit them in the GUI
- Reads and writes configuration files (ConfigObj¹ format)
- Can be bundled with or kept independent from your task²
- Is data-driven and parameter-type aware
- Is 100% Python, using Tkinter and ConfigObj
- Can be an application or a dialog box; returns a Python dict
- Is ready to use! Can be run³ with:
 - any existing cmd-line task in ANY language
 - any existing Python task with a Python API



"collision_predictor" is a fictional example task. TEAL can easily wrap your own program.⁴

display - edit - validate - save - load - set to defaults - type-check - rule-check - run triggers - execute!

```
Terminal - tch - 96x22 - #1
"collision_predictor.cfg" [readonly] 171, 808C

_task_name_ = collision_predictor PREDICT DANGEROUS ASTEROID/COMET CROSSINGS
source = Catalina Sky Survey Source of collision detection data
node = localhost Where to execute? name of node/cluster/cloud

[PREDICTOR - CALCULATION LIMITS]
num_to_find = 78 Stop after predicting this many collisions
years = 45 Predict this many years into the future
least_likely = 2.5 Ignore hits below this % chance of impact

[COLLISION - ESTIMATE DAMAGE]
damage = True Estimate damage for each threat?
units = Torino Output scale (Torino/Palermo/megatons)
map_it = True Generate estimated damage map?
service = Google Internet map service upon which to overlay
least_threat = teal Ignore hits below this Torino threat level (white/yellow/orange/red)
notify = False Notify public in the event of a disaster?
-
```



Parameter choices are saved in .cfg files.

Your task stores parameter type information in a single .cfgspc file. This would typically be installed with the task's package.



```
Terminal - tch - 117x22 - #1
"collision_predictor.cfgspc" [readonly] 181, 1381C

_task_name_ = string_kw(default='collision_predictor', comment='PREDICT DANGEROUS ASTEROID/COMET CROSSINGS')
source = string_kw(default='Catalina', comment='Source of collision detection data')
node = string_kw(default='localhost', comment='Where to execute? name of node/cluster/cloud')

[PREDICTOR - CALCULATION LIMITS]
num_to_find = integer_kw(0, 10000, default=5, comment='Stop after predicting this many collisions')
years = integer_kw(default=100, comment='Predict this many years into the future')
least_likely = float_kw(default=2.5, comment='Ignore hits below this % chance of impact')

[COLLISION - ESTIMATE DAMAGE]
damage = boolean_kw(default=True, triggers='section_switch', comment='Estimate damage for each threat?')
units = option_kw('Torino', 'Palermo', 'megatons', default='Torino', comment='Output scale (Torino/Palermo/megatons)')
map_it = boolean_kw(default=True, comment='Generate estimated damage map?')
service = string_kw(default='MapQuest', comment='Internet map service upon which to overlay')
least_threat = option_kw('white', 'teal', 'yellow', 'orange', 'red', default='yellow', \
                        comment='Ignore hits below this Torino threat level (white/yellow/orange/red)')
notify = boolean_kw(default=False, comment='Notify public in the event of a disaster?')
```

1 - Foord & Larosa, <http://www.python.org.uk/python/configobj.html>
2 - TEAL is available for download within stsci_python/pytools: http://www.stsci.edu/resources/software_hardware/pyraf/stsci_python
3 - TEAL runs on any operating system where Python can be found.
4 - E.g. TEAL is in use at STScI with the latest version of the "Multidrizzle" task.

