**USER Manual**



**open calculation environment**

**text and equation processing**

**report formatting**

**full-text database search**

**v0.9**



|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **overview:**  **source code:   database:**  **forum:** | | | | | |  | <http://structurelabs.com/once> <http://on-c-e.github.io>  <http://on-c-e.net> <http://zero-construction-productivity-growth.net> | | | | |
|  |  |  | | | | | |  |  | |
|  |  |  | | | | | |  |  | |
|  |  |  | | | | | |  |  | |
|  |  |  | | | | | |  |  | |
|  | | | |  |  |  | |  | | |

**on-c-e** is a [**StructureLabs**](http://www.structurelabs.com) open source project that simplifies the process of writing and collaborating on engineering design calculations – the basis of all large successful structural systems. Calculations are produced in three steps:

**write model**

Run **on-c-e** to generate single calcs or organize many calcs into a formatted project calculation report.

**run once**

**add to database**

**CONTENTS**

[1. Introduction 5](#_Toc414413882)

[2. Calc example (no LaTeX input) 7](#_Toc414413883)

[3. Calc example (with LaTeX input) 15](#_Toc414413884)

[4. on-c-e overview 16](#_Toc414413885)

[5. Files 18](#_Toc414413886)

[6. Operations - overview 20](#_Toc414413887)

[7. Operations - syntax 22](#_Toc414413888)

[8. Databases 31](#_Toc414413889)

[9. Resources 32](#_Toc414413890)

[10. Diagrams 33](#_Toc414413891)

[11. SymPy, reST, and LaTeX 33](#_Toc414413892)

[Appendix A - Write a UTF calc using Pyzo 35](#_Toc414413893)

[Appendix B - Add a template to calc-db 36](#_Toc414413894)

[Appendix C - Write a calc in the cloud 37](#_Toc414413895)

[Appendix D – Write calcs using Komodo 40](#_Toc414413896)

[Appendix E – Evaluate calc equations interactively 44](#_Toc414413897)

[Appendix F – Generate PDF calcs on workstations 46](#_Toc414413898)

[Appendix G – Write UTF calcs on Android and iOS 47](#_Toc414413899)

[Appendix H – Minimum Programs and Libraries 50](#_Toc414413900)

[Appendix I – Licenses 51](#_Toc414413901)

[Appendix J – Security and Namespaces 53](#_Toc414413902)

[Appendix K – Examples (separate document) 54](#_Toc414413903)

[Index 55](#_Toc414413904)

Figures

[Figure 1. Individual and project calcs 7](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139560)

[Figure 2. Calc input (with Komodo with syntax coloring) 8](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139561)

[Figure 3. UTF calc output 9](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139562)

[Figure 4. PDF calc output 11](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139563)

[Figure 5. PDF calc output 13](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139564)

[Figure 6. PDF calc output 14](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139565)

[Figure 7. Primary inputs and outputs 18](#_Toc414139566)

[Figure 9. Database Schemas sqlite (left), mongoDB (right) 31](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139567)

[Figure 10. Dia screenshot 33](#_Toc414139568)

[Figure 11. Texmaker screenshot 34](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139569)

[Figure 12. IEP screenshot 35](file:///C:\Users\rhh\Dropbox\_StructureLabs\once-utils\once_manual\once-manual-spec-054_x.docx#_Toc414139570)

[Figure 13. Wakari – terminal and editor in split windows 38](#_Toc414139571)

[Figure 14. PythonAnywhere – side by side browser windows 39](#_Toc414139572)

[Figure 15. Komodo interface 40](#_Toc414139573)

[Figure 16. Screenshots of model equations in IPython shells 45](#_Toc414139574)

[Figure 17. Screenshot of onceutf calc on iPhone 47](#_Toc414139575)

[Figure 18. DroidEdit screenshot of onceutf calc on Android tablet 49](#_Toc414139576)

[Figure 19. http://opensource.org/licenses/MIT 51](#_Toc414139577)

[Figure 20. http://creativecommons.org/publicdomain/zero/1.0/ 52](#_Toc414139578)

Tables

[Table 1. Comparison of oneutf.py and oncepy **Error! Bookmark not defined.**](#_Toc405761631)

[Table 2. Open Source components 17](#_Toc405761632)

[Table 3. Operations summary 21](#_Toc405761633)

[Table 4. File and folder structure 19](#_Toc405761634)

[Table 5. Minimum required programs and libraries 50](#_Toc405761635)

# Introduction

Structural calculations and drawings are at the heart of any design and construction project and their clarity, completeness and responsiveness to changed conditions can have a significant impact on project cost. Developments in open source software over the last decade have provided effective tools that can be applied to improve foundational construction document productivity.

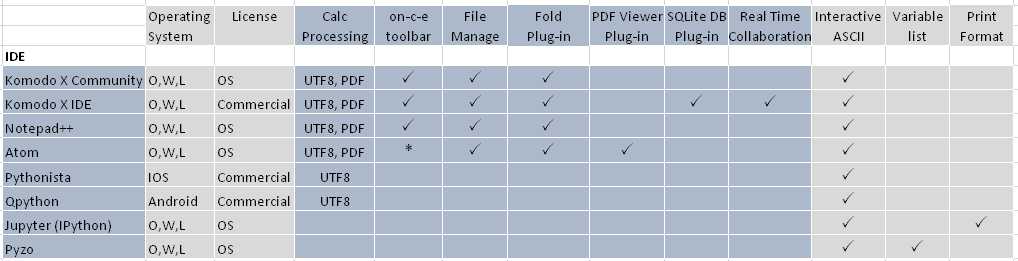
**on-c-e**  (pronounced ‘once’) implements a simple, text based markup language and program for publishing and sharing searchable structural engineering calculations. It is built on open source tools and programs in the following areas:

- Interactive Development Environments (IDE)  
- Python and its scientific libraries  
- LaTex distributions  
- Local and cloud database and file systems

referred to as **calcs**. Because calc input is written in text they are easily stored and searched in a database. Because it generates formal LaTeX output, calcs can be used for reports and building permit documentation. **on-c-e** was started as part of a research and development effort to address factors contributing to the decades-long slide in construction productivity. Calc inputs generate two searchable output formats; **UTF-8** and **PDF**. A UTF-8 calc is instantly generated and is intended for rapid design checks and iteration. It can be edited, annotated and inserted in other documents. The same calc input can also produce PDF calcs formal reports and construction documents. They include graphics and LaTeX math and.

**on-c-e** writes formatted UTF-8 and PDF calcs. [Sections](#appendix_a) 2 and 3 show example calc inputs and outputs. Additional examples are provided in Appendix J and they can be browsed in the online database at [on-c-e.org](http://www.on-c-e.org). Compared to Mathematica ™, Mathcad ™ or MATLAB ™, **on-c-e** reduces the effort needed to publish and share calculations and insure that older models always run. Compared to Excel ™ and hand calculations it improves legibility, search, review, navigation, organization and re-use.

Calcs may also be interactively processed within Python interactive shells and notebooks including [IEP](#section_3), [Komodo Edit](#appendix_c)  and[IPython](http://ipython.org/). Multiple related PDF calcs may be organized in a **project calc** with title blocks, page numbers, table of contents and links.



**Table 1. Platform capabilities**

**Calc inputs** are prepared by typing equations and explanatory narrative in any IDE or text processor using ASCII text**.**  The user controls calculation flow and formatting by inserting single letter markup tags in the text (Table 4).

This manual describes **on-c-e** design objectives, syntax and how to use it on all major operating systems and devices including Windows, OS-X, Linux, Android, and iOS. The interface is a text processor or Interactive Development Environment (IDE) on workstations, a web browser or IDE on networks**,** and an app on mobile platforms. Instructions for using the program on different platforms are provided in the appendices

The half-dozen tags (**[ r ] [ i ] [ v ] [ e ] [ t ] [ s ]**) control calc formatting and interaction with external programs and files. **on-c-e** knows how to handle text, equations, units, significant figures, tables, functions and figures

APPENDIX A - GENERATE CALCS USING PYZO

APPENDIX B - ADD A TEMPLATE TO CALC-DB

APPENDIX C - GENERATE CALCS IN THE CLOUD

APPENDIX D – GENERATE CALCS USING KOMODO

APPENDIX E – GENERATE CALCS INTERACTIVELY

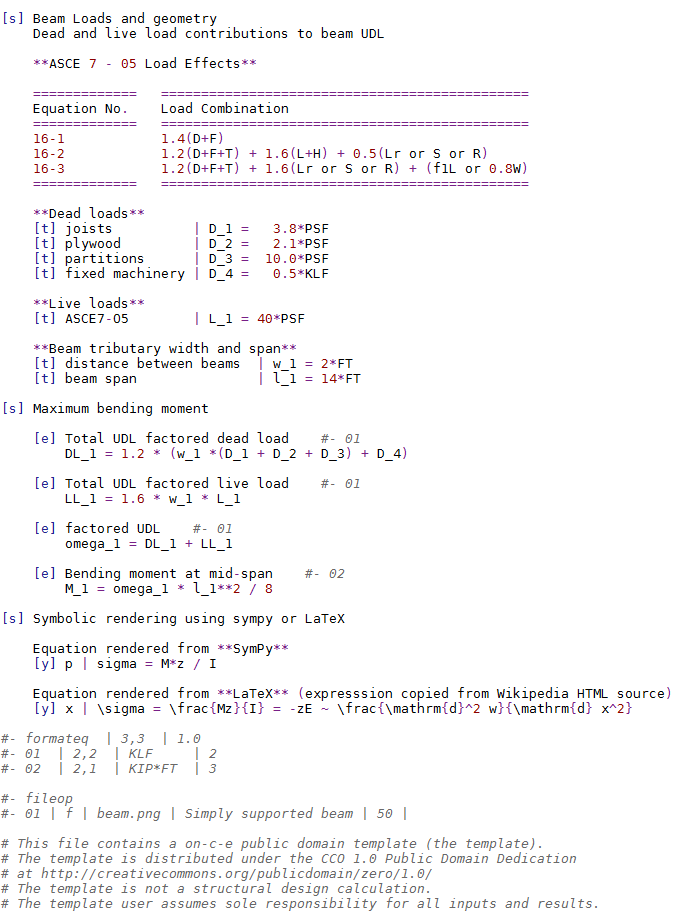
APPENDIX F – GENERATE CALCS ON WORKSTATIONS

APPENDIX G – GENERATE CALCS ON ANDROID AND IOS

**Figure 1. Individual and project calcs**

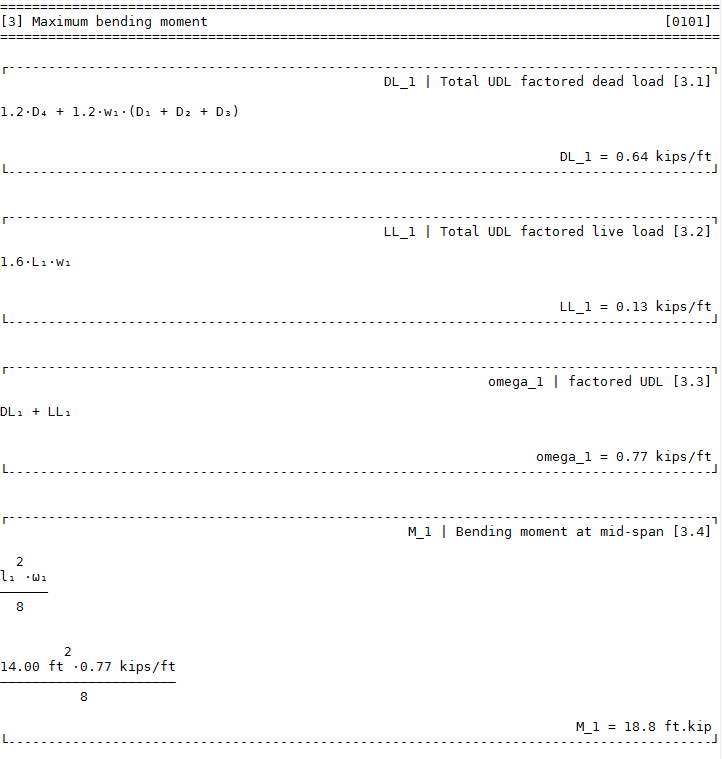
# Calc example (no LaTeX input)





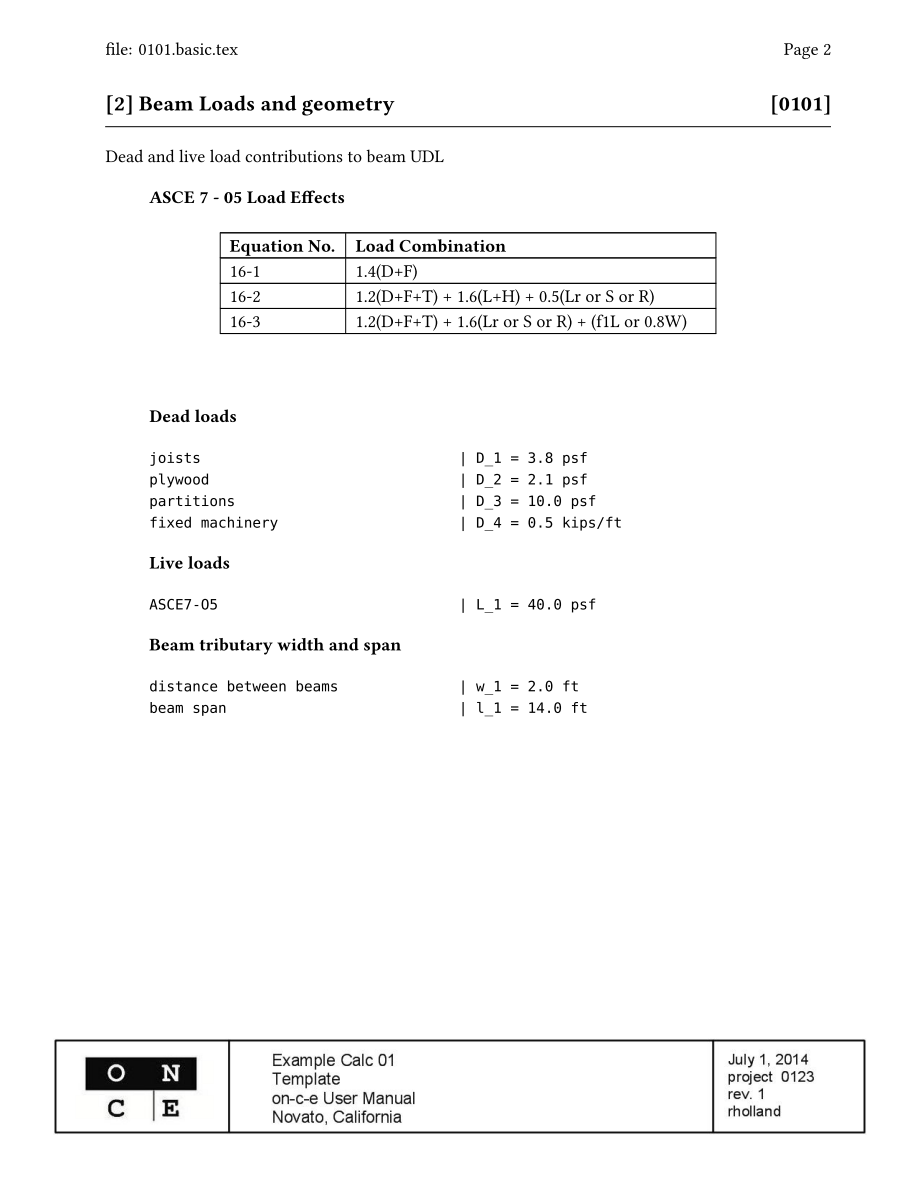
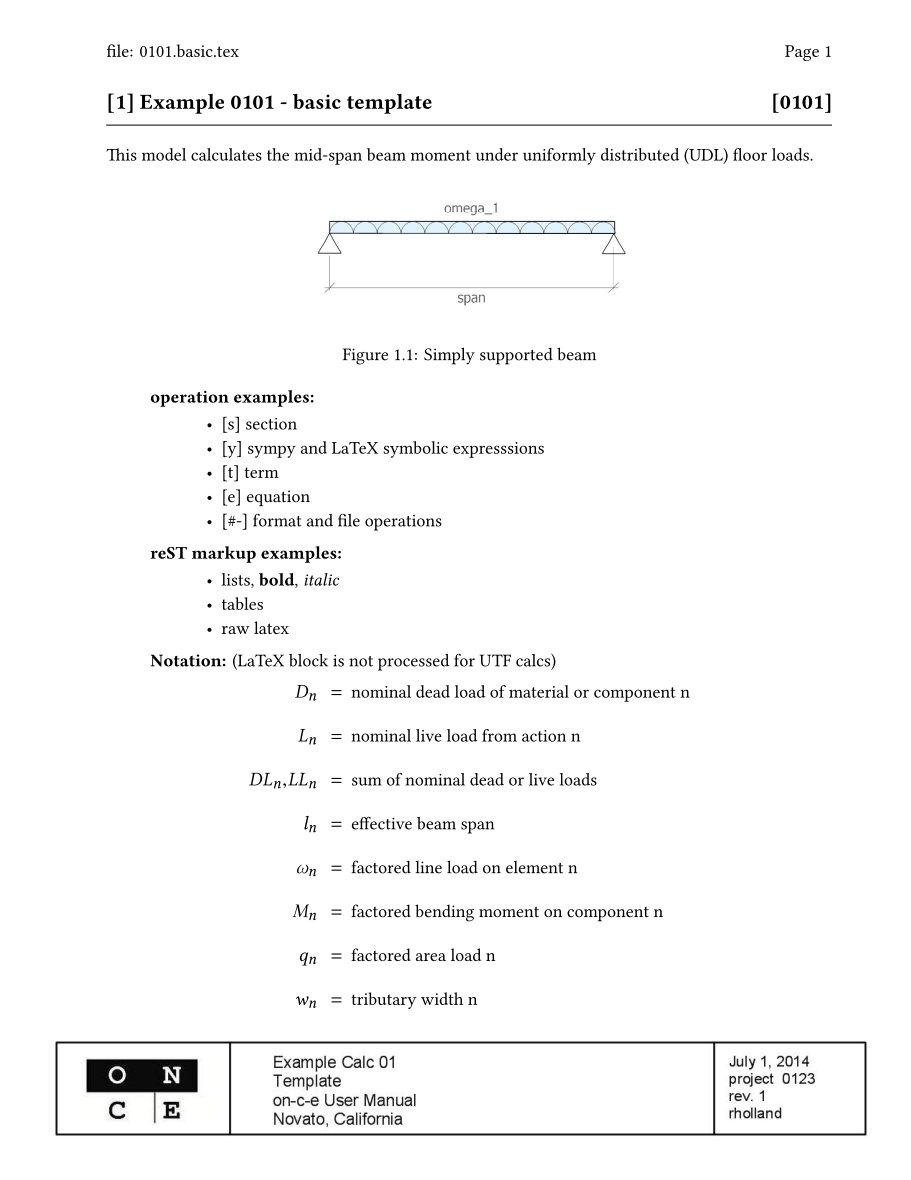
**Figure 2. Calc input (in Komodo IDE with syntax coloring)**

****



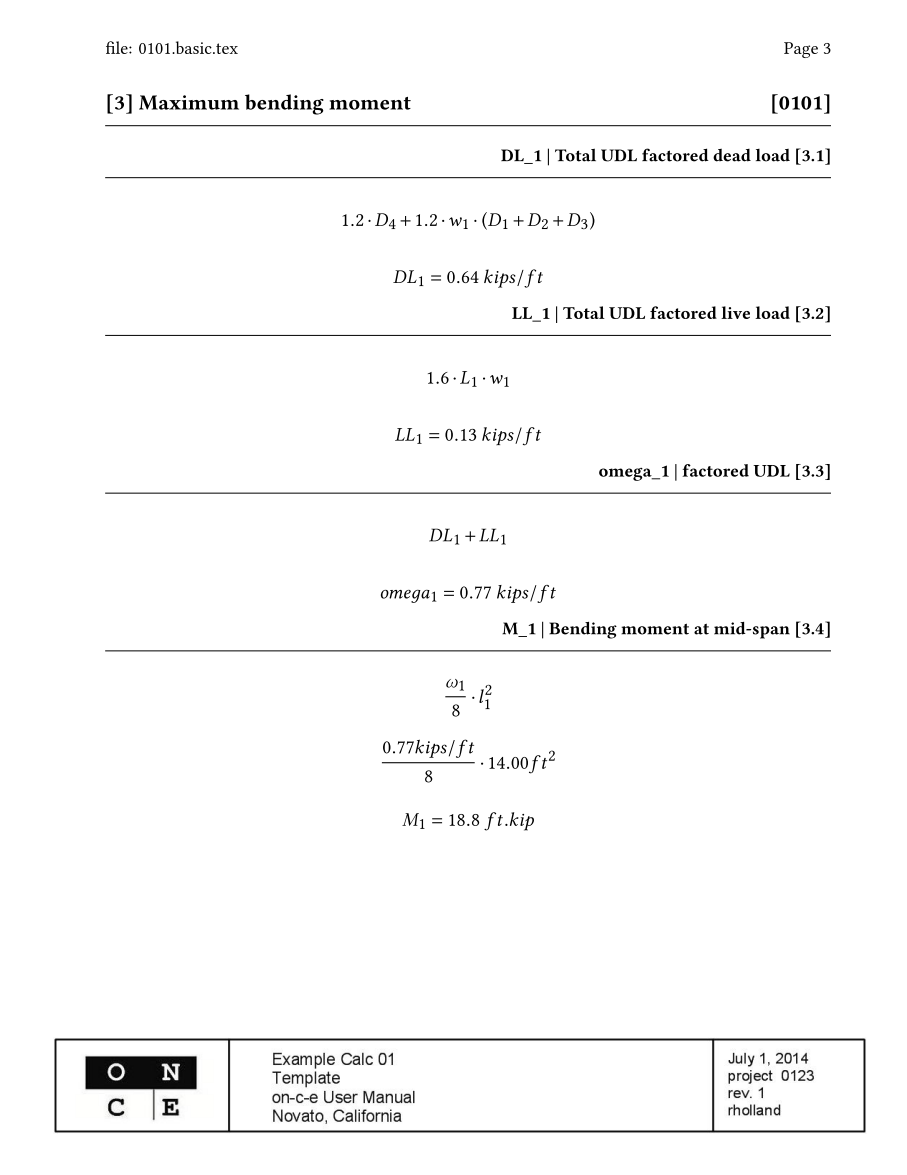
**Figure 3. UTF calc**

**Figure 4. UTF calc (continued)**



**Figure 5. PDF calc output**

**Figure 6. PDF calc output**



**Figure 7. PDF calc output**

# Calc example (with LaTeX input)

# **on-c-e** overview

**Program Objectives**  
- automate calc formatting, including symbolic representations, arithmetic and algebra  
**- make it easy to search and share the program and calcs**- stay out of the way of thinking and writing  
**- produce clear, organized documents  
- run on workstations, mobile devices and in browsers   
- integrate with Python and IPython engineering and scientific libraries**- develop a general tool that can replicate single purpose software tools **- integrate with other structural engineering programs  
Design Approach  
- produce UTF-8 results for fast design iteration and platform simplification  
- produce LaTeX results for clarity, expressiveness and formal documents  
- use a simple, readable, loosely structured markup language  
- build on mature, open source platforms and interfaces  
- use online databases for calc template sharing**

The **OpeN Calc Environment** is defined by **on-c-e** modules, calc inputs, the calc database and Open Source components. **on-c-e** parses calc inputs into dictionaries of equations and operations and processes them through the **Python** **numpy** and **sympy** scientific libraries to produce **UTF** and **reStructuredText** files. The **reST** file is further processed through **docutils**, **LaTeX** and **PyPDF2** libraries to produce **PDF calcs**. Typical calc inputs are a few hundred lines of text and can be stored and searched in many types of databases (see [calc-db](https://structurelabs.knackhq.com/oncedb#home/)). Calc inputs may be combined with other inputs, files and results from external structural engineering programs.Templates are shared inputs with public domain licenses.

**on-c-e components:**

* **ffcc.input.txt:** calc input or template file that produces a **calc**.
* **oncepy:** Python package installed in the site-packages directory of Python by a simple folder copy.
* **onceutf.py:** single file Python module produces **UTF calcs** and simplifies exploration and testing.
* **calcipy.py**: module updates the **IPython** history database with **on-c-e** equations.
* **calcsqlite.py**: adds calc input to a **calcdb** SQLite database
* **calcdb-n.sqlite3 and calc-db:** local and online calc databases.

**Open Source components (except where noted):**

* **Anaconda 1.9.2:**  scientific Python distribution for Windows, Linux, OSX.
* **Pyzo:** Python IDE with integrated introspection, interactive help and portable installation.
* **IPython:** Interactive shell and notebooks for Python and Julia.
* **Komodo Edit / Komodo IDE (commercial):** multilanguage IDE for Windows, Linux, OSX.
* **Wakari:** web based scientific Python distribution with browser interface.
* **DejaVu Fonts:** complete UTF-8 math fonts.
* **Tex Live:** LaTeX distributionrequired for LaTeX and PDF calcs.
* **Dia:** vector drawing program
* **SQLite and SQLiteStudio:** database and front end manager
* **Pythonista (commercial):** Python platform for iOS runs **onceutf.py.**
* **QPython (commercial):** Python platform for Android runs **onceutf.py**

For comparison of Python with other numerical computing environments, see:

<http://www.stat.washington.edu/~hoytak/blog/whypython.html>

<https://sites.google.com/site/pythonforscientists/python-vs-matlab>

**Table 2. Open Source components**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **Komodo Edit IDE** | **Anaconda Python Platform** | **C:\Users\rhh\Dropbox\StructureLabs\admin\logos\once_logos\c008x.jpgoncepy once-db onceutf.py  onceipy.py** | | |
| **Links** | [**activestate.com**](http://www.activestate.com/komodo-edit/downloads) | [**anaconda**](https://store.continuum.io/cshop/anaconda) **(ver. 1.9.2)** | [**on-c-e.org**](http://on-c-e.org/) | [**oncedb**](http://structurelabs.knackhq.com/oncedb#home/) | |
| **OS** | Windows, Linux and OSX | | | | |
| **Details** | Full featured open source IDE for multiple languages | Enterprise-ready distribution for scientific computing | Python package for writing structural calculations | | Online template database with web interface |
| **Sponsor** | Active State  Canada | Continuum USA | StructureLabs USA | | |
| **License** | Mozilla Public License | Various Open Source Licenses | Program: MIT License Models: CCO1.0 Public Domain | | |

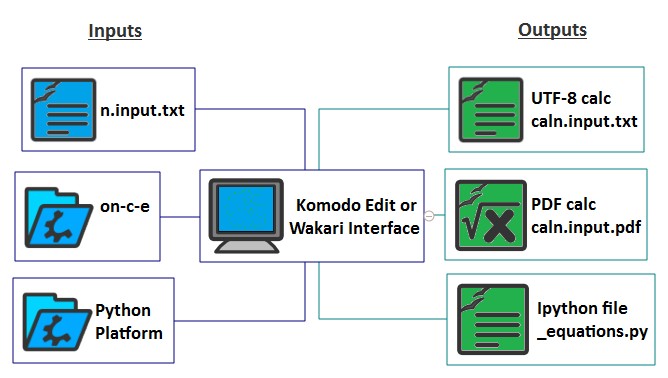
# Files

The fundamental **on-c-e** component is a calc input file **ffcc.name.txt**, where **ffcc** is the calc designation, **ff** is the two digit folder number, **cc** is the two digit calc number and **name** is a user-created calc name. The calc designation is used for calc and project organization and must be unique in a project. Calcs are stored in folders which in turn are stored in a project folder (Table 4). Standalone calcs do not require a folder structure but do require a leading calc designation. When a calc is run it produces the following output files in its folder (Figure 3, Table 2):

* ­­UTF-8 calc **ffcc.name.calc.txt**
* Optional PDF calc **ffccc.name.calc.pdf** and temporary files (removed by default)
* IPython input file **\_equations.py** for interactive analysis and database summaries
* Execution log file **\_calclog.txt**, also (partially) echoed to the terminal during execution
* Optional .rst, .tex, .log, .out, fls, fdb\_latexmk – see Appendix F
* Optional project calc **project.pdf** assembled from a specified set of division folders

Related calcs may be grouped in division folders. Calcs may incorporate other calcs in the project, external functions, data, and batch or script control files. Externally referenced files (i.e. figures) must be located in the calc folder, and external calcs must be located in a project division folder. External files are identified by file name only. The program recreates the full path name internally, which allows division folders to be moved to new projects without changing the calc. The exceptions include those files processed by the file operation options r (read) and e (edit). These two operations typically use files produced by other programs with their own preferred file structure. To work with these files in their natural location a full path name is required (see [file operations](#fileop), Appendix B).

A calc folder has the name **ff.foldername** where **ff** is the unique folder number. Each project folder may contain up to 100 calc folders each with 100 models using unique numbers 00 to 99. The project folder contains an optional project definition file that defines the structure of the project calculation documents. It may contain the project wide unit and PDF style definition files (**unitc.py**, **once.sty)** that override program defaults. If unit and style definition files are also included in a calc folder they will override both the project folder and built-in defaults for those calcs.



**Figure 8. Primary inputs and outputs**

|  |  |  |  |
| --- | --- | --- | --- |
| Project Files | | Project / Calc Folder Structure | Calc Files |
| **ddmm.input.txt** input |
|  | **(project.txt)**  Project file |  |
| **(background.pdf)**  Calc background | | **(once.sty)** PDF style |
|  | | **ddmm.name.calc.txt** UTF-8 calc |
| **(ddmm.name.calc.pdf)** PDF calc |
| **(project.pdf)** Project calc | | **equations.py** Python equation file |
| **(project.log.txt)** Project execution log | | **\_calclog.txt** Calc execution log |
|  | | Notes: 1. Input files are in blue; output in green 2. Optional files are in parenthesis |  |

**Table 3. File and folder structure**

# Operations - overview

There are six **on-c-e** markup tags used to produce calcs:

**[r]**ublish

**[i]**ection

**[v]**erm

**[e]**quation **[t]**rray

**[s]**rray

Tags are processed in the order entered. During processing, sections and equations are automatically numbered and formatted. Equations are rendered both symbolically and with numerical substitution, similar to a hand calculation. 1D and 2D tables are generated from equations, imported data and spreadsheet-like formats. Units and decimals are controlled at the equation level by format statements positioned at the end of the calc input to improve readability. [Table 3](#table3) and [Sec](#figure3)tion 7 provide further details.

A bracketed **tag** (**[]**) specifies printable content. Hash-dash tags (**#-**) tags specify formatting and file processing operations. Operations may occur in any order, without nesting, as long as each variable or parameter was previously defined. Operation parameters are separated by vertical bars (**|**). Default parameters may be omitted but the bar delimiters (with at least one space between) must be provided. Tags include **[s]**, **[y]**, **[t]** (single line operations) and **[c]**, **[a]**, **[f], [e]** (multiple line operations terminated with a blank line).

Tables (arrays) can be manually configured like a spreadsheet or generated using equations and range variables. If variables with units are used in a generated array they are stripped before evaluation.

The **file operation tag** **#- nn** designates a file operation definition number that includes the options **b, r, o, w, n, i, e, s**. For example, the option parameter **|s|** processes a Python file or script that defines function imported into defines functions available to the model. A loaded function is processed in the model by the **function tag [f]**. The option **|i|** imports a comodel into the main model. A comodel is an imported model that co-executes with the main model, but does not itself import other models (only one level of model import is permitted). Other options import figures and text and execute read, write, edit and run operations on external files and program (PDF calcs insert figures; UTF calcs insert references)

Any text in the model not associated with an operation tag is passed through to the output unchanged. **reStructuredText (reST)** markup is retained and used when processing PDF calcs. For example, surrounding text with \*\*double asterisks\*\* will print **bold**, \*single asterisks\* will print *italics* and a vertical bar at the beginning of a line will force a new line. **reST** directives, i.e. raw latex, are also processed by PDF calcs but not echoed in UTF calcs. See the [reST reference](http://docutils.sourceforge.net/docs/user/rst/quickref.html) for further documentation.

Project operations use tags **[p]** , **[n]**. Project calcs are grouped sets of numbered **PDF calcs** with a title page, table of contents, title blocks and navigational links. The **[p]** operation writes a project calc with the name **project\_file\_name.pdf** to the project folder, overwriting any existing file. The project operation is divided into a project data definition section and a calc folder inclusion list (Table 4 and [Appendix B](#appendix_b)). Project defaults can be overridden in each calc. Project data is defined by a dictionary of keyword-data pairs. The project format operation **[n]** defines the page print location for specific project data.

|  |  |  |
| --- | --- | --- |
| **Tags** | **Operations (optional parameters)** | **Notes** |
| **[[r]](#disk)** | Run  [r] os | *description* | *ok* | *decimal  script* | *expression* | *operator* | *limit* |  |
| [**[i]**](#array) | Insert  [a] *figure*| *table description  function* | *variables  model* | *variables  text* | *variables  read* | *variables  write* | *variables* | For format reference numbers see format operation. |
| [**[v]**](#equation) | Value [v] block title  *description | var = value* | Values are defined in constants, typically with units |
| [**[e]**](#equation) | Equation  [e] (*equation description) #- format  var = expression* | For format reference numbers see formateq operation. |
| [**[t]**](#term) | Table  [a] *format\_number* | *table description  range variables  var = expression* | Vectors or tables |
| [**[s]**](#section) | Section [s] *section heading (#- toc)* | *#- toc* inserts a list of sections |
| [**#**](#d_op)**- nn** | File  #- *fileop reference number (file description)* | nn is a 2 digit file operation ref number |
| [**#- page #- stop #. comment**](#comment) | #- page #- stop #. *comment* | #page inserts page break. #stop terminates processing. |
| [**[p]**](#project) | Project data [p] *(pdf size)* | *(background.pdf) keyword = data*  *keyword = data  01\_divisionfolder* | *(division title)*  *model number* | (pdf size) | (background1.pdf)  *keyword, format*  *keyword, format   model number* | | | *keyword, format  02\_divisionfolder* | Each model file is processed to a PDF if a PDF does not exist. |
| **[[f]](#pformat)** | Project formats  [n] *format* |[n] format | | Terminate each format with a blank line. |
| **[[k]](#pformat)** | [k] *keyword* | *x location* | *y location* [k] *keyword* | *x location* | *y location* |  |

**Table 4. Operation summary**

# 7. Operations - syntax

The calculation structure and equation formatting is controlled by **on-c-e** operation tags. **reST** directives and inline markup may be inserted between tags. The calc operations and associated tags are summarized in this section (also see [Table 3](#table3)). Operations that include printable content use bracket tags ([]). Operations that direct processes, i.e. formatting and file operations, use hash-dash tags (#-).

**single line:** [s]**,** [y]**,** [t], #- nn

**multiline:** [c]**,** [a]**,** [f]**,** [e] Multiline operations terminate with a blank line.

**formatting:** #- formateq and #- fileop

**project:** [p]**,**  #- pformat

File operations (#- file) include the options w, r, i, t, e, f, o, s. The mnemonic **syt café** **write fos** (sit café, write FOSS) may help to remember tag options   
(FOSS is an acronym for “free and open source software”).

**reStructuredText (reST)** is used for formatting words, lists and tables in PDF files.Commonly used reST markup with additional requirements related to **on-c-e**, include:

|  |  |
| --- | --- |
| \*word word\* | Render italics |
| \*\*word word\*\* | Render bold |
|  |  |
| List heading  - line 1  - line 2 | Bulleted lists must be indented at least 8 spaces from the left edge (two 4-space tabs). The list must also be indented relative to a list heading. |
|  |  |
| ======== =========  Column1 Column2  ======== =========  data 1 data 2  data 3 data 4  ======== ========= | Tables must be indented between 0 and 7 spaces from left edge. |
| ..raw:: latex  `  \some{latex}  ` | Within a reST directive use a diacritic in the first column to specify the beginning and ending line break. Directives are not printed or processed for UTF calcs. |

Refer to online reST documentation for additional details.

<http://docutils.sourceforge.net/docs/user/rst/quickref.html>

**Note: To avoid name collisions use subscript notation (i.e. D\_1, D\_2) for short variables names. To improve model readability, input within sections ([s]) should be indented (4 spaces is standard).**

**[****s]** section description

The section operation organizes the calculation. Formatted sections are labeled on the left with an incrementing section number and description, and on the right with the model number. Equation numbers are generated within sections as **s . e** where **s** is the section number and **e** restarts at 1 in each new section. In PDF calcs, sections start new pages. Indent text and operations (4 spaces) within sections to improve model legibility (see [Appendix A](#appendix_a)).

section description: single line heading

**[y]** param | LaTeX, Python or sympy expression

The symbolic operation formats a symbolic expression without evaluating and renders it in UTF and LaTeX form. Expressions may continue over multiple lines. param is one of:

s:Sympy expression rendered in UTF and PDF calc

p: Sympy expression rendered in UTF and PDF calc and to a file.

x:LaTeX expression is rendered to PDF calc and to a file.

where the automatically generated file name for the rendered equation image is latex[eqnumber].png. The p and x options require a LaTeX installation.

**[t]** description | variable = expression

The term operation is used to assign values to terms used in equations. It evaluates statements with constants. In a list of terms, put terms with vectors at the end for improved formatting.

description:term description

variable: any variable that meets Python naming conventions  
expression:math expression that uses constants

**#-** nn (description)

The file operation designation specifies the file processing and formatting definition number. See [file operation](#file)  for further details.

nn: two digit integer that matches the reference number of a file definition operation

description:brief file operation description

**[****c]** description | ok | decimal, decimal

expression | operator | limit

The check operation checks an expression against a limit for the specified operator.

description:description of comparison

ok:printed phrase if compare evaluates true; prepended by **not** if false:

decimal:number of printed decimal places in equation and result (default 3, 3)

expression:expression to evaluate – typically a ratio

operator:Python comparison operator (<, >, <=, >=, =)

limit:value or expression that evaluates to set upper or lower limits

**[a]** array description #- format ref  
 range variable1

range variable2 (only for for 2D arrays)

var = expression; or array reference; or spreadsheet

The array operation input has three different forms.

1. The first form is similar to the equation operation but generates 1D or 2D tables with column and row labels. Range variables and expressions must be unitless. The program attempts to remove units before processing. If only one range variable or vector is provided a vector table (single row) is output.

variable1:a unitless range statement or array variable  
variable2:a unitless range statement or array variable for 2D tables.  
var = expression:unitless equation using Python syntax

1. The second form is a Python list of lists which operates like a spreadsheet

**var = [**

**[‘header1’, ‘header2’, ‘header3’ ],**

**[ 1.1, 2.2, ‘\_rc[1][0] + \_rc[1][1]’],**

**[ 3.3, 4.4, ‘\_rc[2][0] + \_rc[2][1]’], ]**

Where:

* any term in quotes is a string
* the first row is composed of column header strings and is evaluated on the first pass. Do not include spaces around any operator symbols in headers.
* any term not in quotes is a number or defined variable and is evaluated on the first pass
* any term in quotes that also has one of + , - , \* , / operators set off with spaces on each side is evaluated on the second pass after the numbers and variables have been evaluated. The prefix \_rc is a built-in function that evaluates references to list elements using the row and column designation.

1. The third form has a single array definition line: var = arary\_var

Where arary\_var is a previously defined or imported array.

**[****f]** function description

function\_name(args) | var

The function operation executes a function and assigns the return value to variable var. The function is imported from a Python file using the #- file operation and s option. Function doc strings are printed. Decimals are controlled by the function.

var :one line description of function  
function description :short description of function

function\_name(args):name of function and arguments to be executed

**[e] (**description) #- format ref  
 var = expression

The equation operation evaluates a Python math statement with units and prints results with varying levels of detail as controlled by the #- format operation. See formatting operation. The expression may extend over more than one line.

The description may be omitted and the equation started on the first line. The first line must include an ‘=’ sign. If a format reference is not provided the first formatting ref in the list will be used as a default.

description:optional equation description or building code reference

format ref:two digit integer that references a formatting operation.

var = expression: equation using Python syntax

**#- formateq** | (default dec, default dec) | (pdf margin)

format num |(dec, dec) | (units/row label)` | (prt code/col label) format num |(dec, dec) | (units/row label) | (prt code/col label)

The formateq operation controls calc output type, printed detail, decimals, units and labels for equations and arrays. Use print detail code 0, 1 or 2 for equations that return vectors. 2D arrays do not use a print detail code. Set the command line option to   
**–noclean** to retain the **rst** and other temporary files when generating a PDF file. See Appendix D.

default decimal: number of printed decimal places in equation and result (default 3, 3)

pdf margin:Bottom PDF page margin in inches. If 0 a UTF calc is output.

format number: Unique five digit equation or array format number between 1 and 99999. Typically the first four digits are the model number and the fifth digit is the section number. This pattern facilities organization and guarantees uniqueness.

deci, deci:override umber of printed decimal places in equation and result

units/row label:unum unit for dependent variable or label for first variable

print code/column label:For equations provide print code (default is 3) where:

0 - evaluate but do not print result

1 - print result

2 - print symbolic expansion and result

3 - print full result: symbolic expansion, substitution and result

For arrays, provide column labels in place of print code.

**#**page

**#**stop

**#** comment

The hash tag is used for non-printing comments. A space is required between the hash and comment. The space is omitted when it is used for non-printing operations.

#page inserts a page break in the PDF calc.

#stop stops model processing at the point it is encountered. This can be useful for developing and debugging models.

# comment inserts a non-printing commen

**#- fileop**

#- nn | option | (path)/filename | (v1) | (v2) | (v3)

#- nn | option | (path)/filename | (v1) | (v2) | (v3)

The fileop operation list processes external disk files using a single letter to specify processing procedure. Each operation has a designated two digit number which is used for reference in the model. Typically files must be located in the model folder and are specified by file name only. Full file paths are only used for the rand eoptions since the files these operations access may be located anywhere in the file system. This approach makes a model as portable as possible across different folder organizations.

**options**

#- nn: file operation designation number

w:write values of a variable to CSV file. Overwrite existing files. w+appends to file.

r:read CSV file data into an array variable; full path required

i:insert and process a comodel file

t:add an external text file (boilerplate) to the output.

e:edit lines of an existing text file and save an edited copy of the file; full path required

f:insert an image from a file into the calc (jpg, png etc.)

o:run an operating system script file. Typically runs a program.

s:run a Python file (script) in model. Typically used to incorporate external functions.

filename:file name used in disk operation. Full path required for |e| and |r|.  
v1-v3:variables specific to each file operation option. See details.

w:write array values to file (uses v1, v2 and v3)

Write contents of array v1 to the division file specified by the filename. Each line or row is written using the **numpy.tofile** method. To append data to an existing file use the option w+. v2 specifies the separation character (comma is default, for a space character use \*).v3 is the data type (default is character format %s).

r:read file data into variable (uses v1, v2 and v3)

**path/filename:** os\_path/filename

Read CSV contents of **path/filename** into variable v1and store as array using the **numpy.genfromtext** method. v2 specifies the separation character (comma is default, for space character use \*). v3 specifies the number of lines to skip before reading the data (default is 0).

i:insert and process a comodel file.

Merge a comodel file from the project into the main model file. The comodel itself cannot include a comodel. Sections and equation numbers are integrated into the main model sequence. The comodel directory is determined by the program from the model number. v1-v3 not used.

t:insert text file contents into output (uses v1)

Insert the contents of the text file from the division folder into the model without processing. The v1 parameter optionally specifies the line range to be inserted as lines n1:n2 inclusive.

e:edit text file (uses v1)

**path/filename:** os\_path/filename

Edit an external file at run time. Lines immediately following the edit operation have the form:

n | replacement text for line n

where nis the line number to be replaced with the specified replacement text. The v1 parameter is appended to the file name before it is saved with edits i.e. if v1is **copy** and the filename is **file.tcl** the edited file is saved as **filecopy.tcl.**

Template variables may be used in the replacement line. A template variable is created by appending a **%** in front of the variable name used in the model.

f:insert figure from file into calc (uses v1, v2 and v3)

Insert figure (jpg, png etc**.**) from division file with caption v1. Figure labels are automatically added and incremented. v2 specifies the image width in percent of document width. The figure is centered and the aspect ratio is maintained. v3 specifies a side by side arrangement for a second consecutive figure using the keyword adjacent. For UTF-8 output the following text is inserted:   
**Figure n: caption - file: filename**

o:run an operating system command or executable file

The filename, accessible from the division folder, is the name of the executable file. v1-v3 not used.

s:execute an external Python script in model namespace

Execute an external Python script from the division folder. Methods (functions) and variables will be processed in the model namespace. v1-v3 not used.

Project operation code has not been released

**[p****]** (project default size) | (default background.pdf)

(keyword = data) | (format\_number)

(keyword = data) |

(keyword = data) |

**.**

**.**

01(\_divisionfolder) | (division title)

01yy.model.txt| size| background2.pdf

keyword = data | keyword = data | | |

02 | (division title)

02yy.model.pdf| size| nobackground

03 | (division title)

03yy.model.txt| size| background2.pdf

keyword = -1| kewyword = -1| | |

04 | (division title)

04yy.model.pdf| | omit

| | | |

04yy.model.pdf| size |

05 | (division title)

The project operation is specified in the project file and provides project calc information. Project files are located in the project folder which in turn contain division folders as subfolders.

size:size and orientation settings of PDF pages.

portrait\_letter (default)

landscape\_letter

landscape\_tableau (11x17)

portrait\_A4

landscape\_A4

background.pdf:user file name for the calc background (default: none)

keyword = data:keyword-data pairs to be added to the project dictionary. Any key word may be defined except for key words reserved for project calc printing. They include:

ptitle, pname, padd1, padd2**,** pstate, pzip, powner, pnumber, eng**,** date, rev, stamp1, stamp2

xx(\_divisionfolder):by default include calcs from this folder in the project calcs. If \_divisionfolder is provided as part of the folder name it is used for the division name in the project calcs (i.e. table of contents). If division title is provided it is used instead.

For each model in the folder, if a PDF calc is available, it is used. Otherwise the model is run to produce the PDF calc.

Project settings may be overridden for a specified calc or model. If default background.pdf is set, then set the value to nobackgroundto suppress the default template and all keywords for a particular calc. To omit a calc from the project set use omit for the template value. A keyword data may be redefined for a specific calc. To suppress writing the keyword value to a calc set its value to -1. To write only keyword data without a template specify the keyword data on the keyword override line.

The project or division folder may also include the file **once.sty** which defines the LaTeX PDF style settings, and the **unitc.py** file which defines calc units. They will override the default settings in the **oncepy** package directory – division folder files override project folder files.

**[n]** format\_number

keyword | x location | y location

keyword | x location | y location

keyword | x location | y location

.  
.

The project format operation describes the print location on a calc sheet of keyword data defined in the project operation. Terminate format number definitions with a blank line.

keyword: project keyword

x location:x location of keyword value on page in points from upper left corner.

y location: y location of keyword value on page in points from upper left corner.

# 8. Databases

🖳

Calcs and projects can be organized and shared as zipped files or in searchable databases. Because calc inputs are text it is relatively straightforward to build searchable databases for sharing and re-use. **on-c-e** currently implements two types of databases (Figure 9). The first type is structured for web browsing and sharing and the second is designed for local use within an office or project.

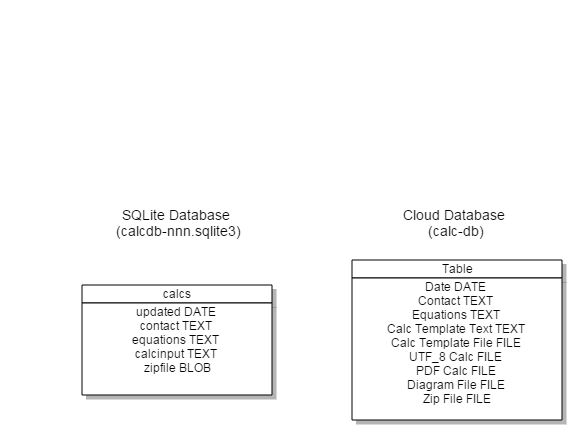
A web-accessible database that uses mongoDB with the schema shown in Figure 4 is available [here](https://structurelabs.knackhq.com/oncedb#home/) and is further discussed in Appendix B. The database and interface is useful for browsing complete calcs and their inputs.

SQLite is used for local database storage. The schema for the local database is similar to the cloud but omits the records that are intended for online viewing and browsing. A convenience script, calcsqlite.py, adds calc input to the local SQLite database. It can be invoked from the command line in the folder containing the calc or through a tool bar button in Komodo. It has the form

python –m calcsqlite\_database\_path calc\_file\_designation [-c contactinfo –z zipfile]

where the terms in parenthesis add the specified information to the database and are optional. Date and equation summary records are added automatically.

**SQLite mongoDB**



**Figure 9. Database Schemas for local processing (left), mongoDB (right)**

# 9. Resources

🖳

##### Share models

**Download Models**<http://on-c-e.org>

**Upload Models**<http://on-c-e.org/upload-model>

🖳

##### Run locally

**Download and install Anaconda 1.9.2.** Using ***pip***, install **Unum**, **tabulate** and **PyPDF2**. Install **Tex Live** 2014. Download and install Pyzo and Komodo Edit (see Section 2 and Appendices D and F).

Although any text editor is adequate for composing and running models, the most effective editor is an interactive development environment (IDE). It provides project, file management and font management, code navigation, templates, custom toolbars and macros, syntax coloring, window layout controls and remote file access. Komodo Edit is a full-featured, cleanly designed, open source IDE which can manage **Python, TCL, Ruby** and other programing languages applicable to **on-c-e** and other structural engineering work. A number of **on-c-e** specific tools and convenience macros are provided for Komodo ([Appendix C](#appendix_c)). Although Komodo is discussed in this manual, other IDEs or text editors may be used.

Commercial support for installing and running the program is available on an annual basis at [StructureLabs](http://structurelabs.com/). It includes phone support and a private **oncepy** installation on a [Webfaction](http://webfaction.com) server that is accessed over SSH and SFTP and will be used for interactive support.

👓

##### Find

About [on-c-e.org](http://on-c-e.org)  
User manual <http://on-c-e.info>

onceutf.py <http://on-c-e.info>  
oncepy <http://on-c-e.org/programs/>  
model database [http://](http://structurelabs.knackhq.com/oncedb#home/)on-c-e.org/  
Code and docs <http://on-c-e.github.io/>  
Road map <http://on-c-e.us/>

Komodo Edit <http://activestate.com/komodo-edit>  
DejaVu Fonts <http://dejavu-fonts.org/wiki/Main_Page>  
Workstation (Anaconda 1.9.2) <http://repo.continuum.io/archive/index.html>  
Web Platform (Wakari) <https://wakari.io/>  
Pyzo <http://www.pyzo.org/index.html>  
TeX Live <https://www.tug.org/texlive/>

# 10. Diagrams

🖳

Diagrams are incorporated into **on-c-e** PDF calcs as bitmaps (png, jpg etc.) using the figure option |f| of the file operation #- file. Structural diagrams are often drawn to scale and can be efficiently reused when described in vector and parametric form that is exported as a bitmap.

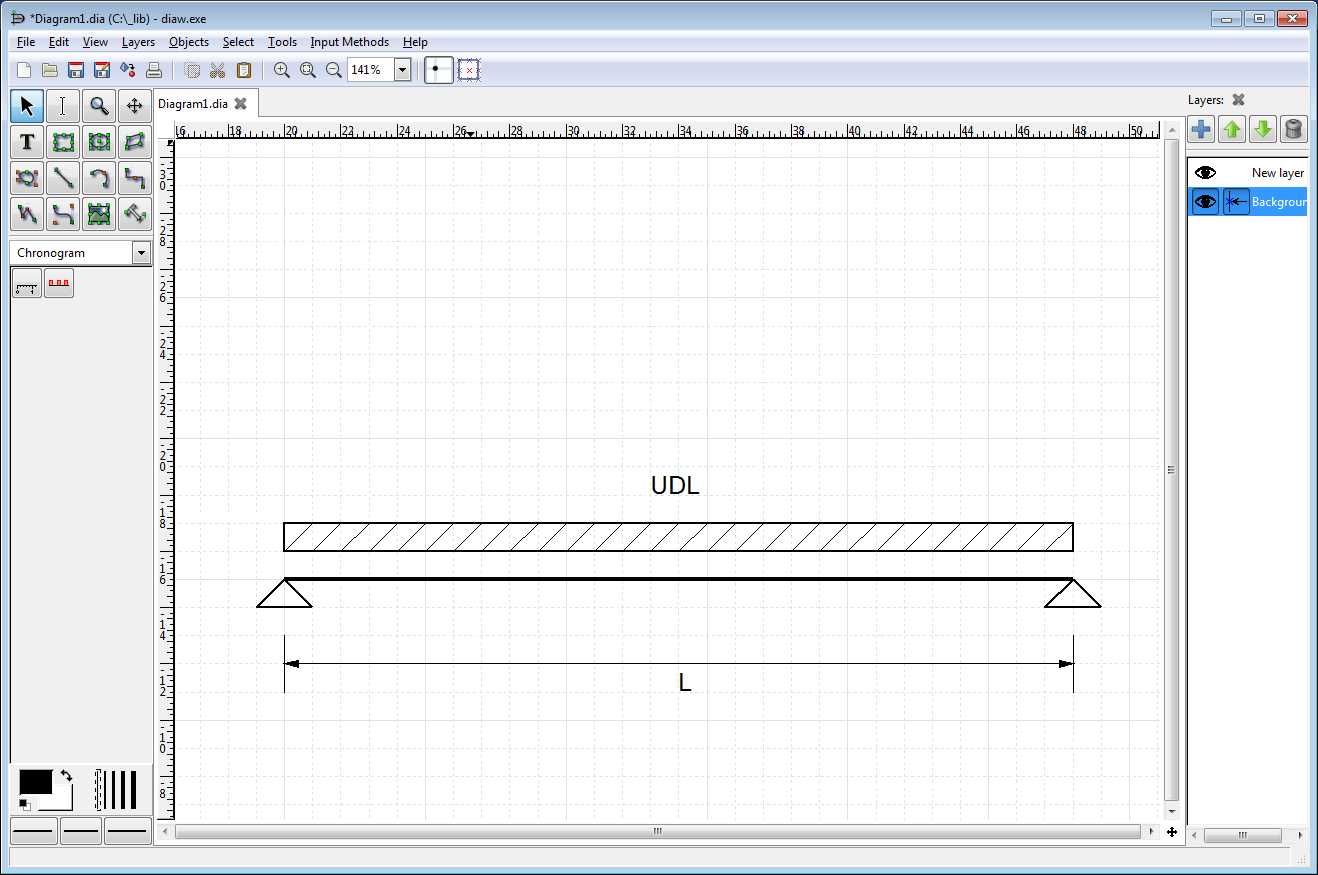
**2D**

The Open Source [Dia Diagram Editor](http://dia-installer.de/) may be used to interactively create and edit vector diagrams. C:\Users\rhh\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\3.jpgIt writes SVG and DXF, the two common scalable vector file formats for 2D diagrams. Parameterized diagrams can be created in both formats using Python scripts and the **svgwrite**, **dxfwrite** and **ezdxf** libraries. Custom shapes for structural engineering can be added to its library. The freely available vector drawing program [**DraftSight**](http://www.3ds.com/products-services/draftsight/overview/)may also be used to produce SVG, DXF and bitmapped drawings.

**3D**

3D diagrams can be created using **SketchUp** or [SketchUp Make](http://www.sketchup.com/products/sketchup-make) (**skp** format), or the Open Source [**FreeCAD**](http://www.freecadweb.org/)(variety of formats). Drawing files can be exchanged between the two programs using the COLLADA **dae** format. The Python **pycollada** library is required for FreeCAD translation (use **pip install pycollada**).

Diagram files (**svg, dxf, skp, dae**) prepared for **on-c-e** templates may be uploaded to a separate field in [**once-db**](http://structurelabs.knackhq.com/oncedb#tab-about/)or included in a template zip folder.



**Figure 10. Dia screenshot**

# 11. SymPy, reST, and LaTeX

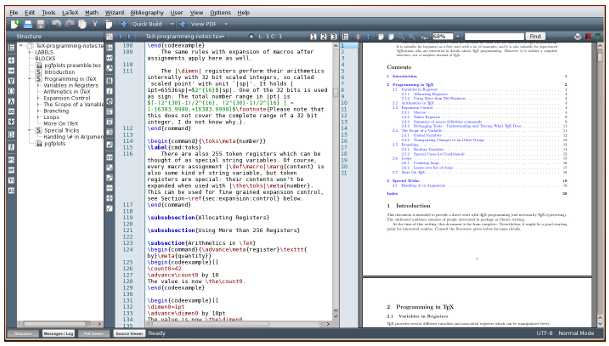
🖳

**on-c-e** C:\Users\rhh\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\6.jpg incorporates LaTeX into calcs through the SymPy and reST libraries. Since most of the LaTeX formating operations for equations are built into the program, LaTeX only needs to be written in on-c-e calcs when it is convenient for explanatory purposes or specific formatting. LaTeX equations are available from a number of sources including Wikipedia articles (use ‘show source’ in browser) and online libraries i.e.

<http://www.equationsheet.com/>  
<http://latexsearch.com/>

They can be pasted into models and rendered using the **[y]** symbolic operation.

[**Texmaker**](http://www.xm1math.net/texmaker/) is useful tool for learning and practicing LaTeX. It is a free, modern, cross-platform LaTeX editor for Linux, OSX and Windows systems that integrates tools needed to develop documents with LaTeX, in one application. Texmaker includes Unicode support, spell checking, auto-completion, code folding and a built-in pdf viewer with syntax support and continuous view mode. Texmaker is easy to use and to configure and is released under the GPL license. The integrated, interactive, interface includes three panes. The left pane shows the document structure, the central pane contains the document template and contents, and the right pane shows the generated pdf.



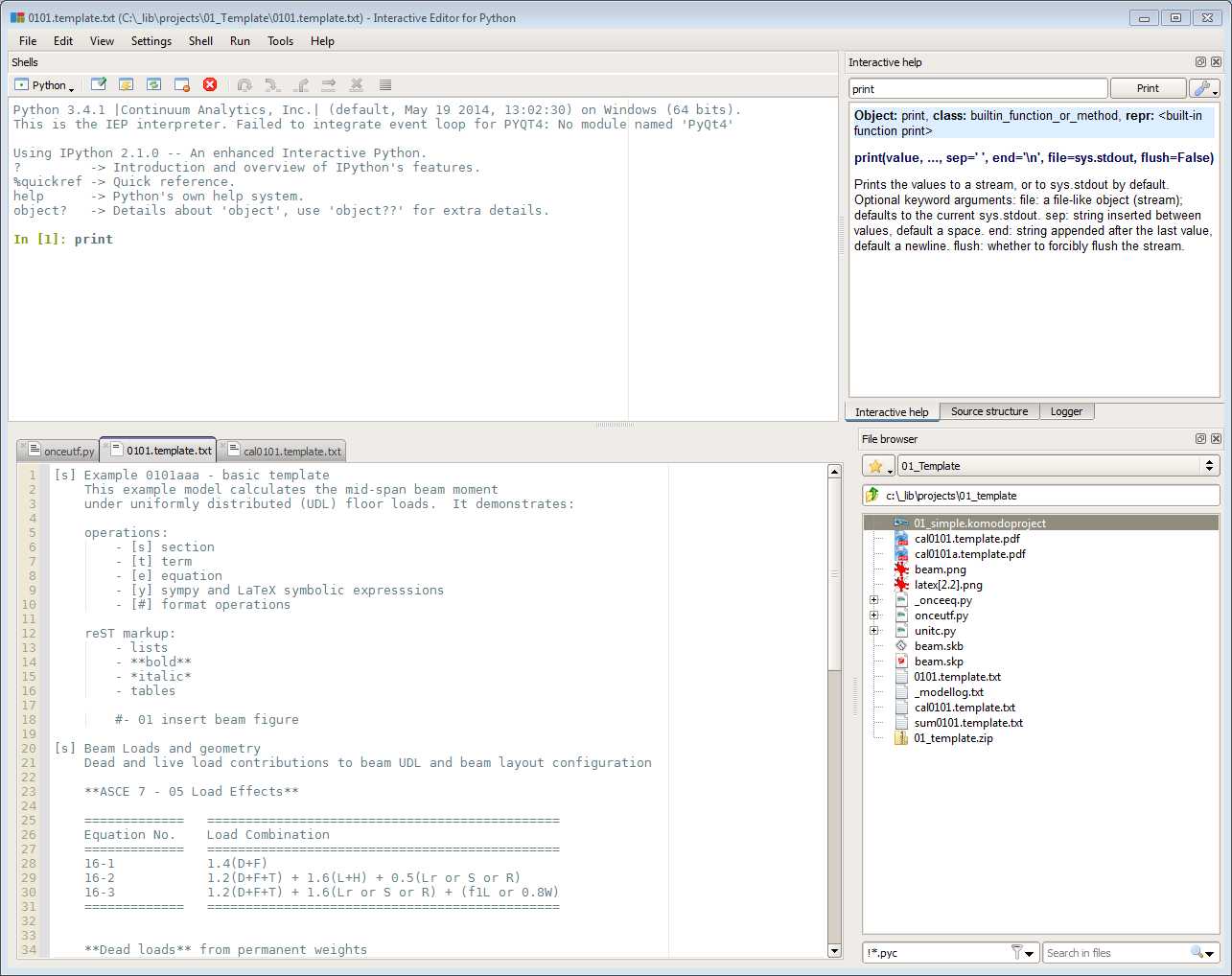
**Figure 11. Texmaker screenshot**

C:\Users\rhh\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\6.jpgAppendix A - Write a UTF calc using Pyzo

[IEP : Interactive Editor for Python](http://www.iep-project.org/) is a self-contained interactive interface for running Python programs on a workstation. It is installed as part of the [Pyzo](http://www.pyzo.org/index.html) science and engineering distribution (based on Python 3). Pyzo can be installed in any folder without administration rights and does not affect existing Python installations. If you want to install a limited Python science platform this is a good placed to start. For a full installation Anaconda (see Appendix D) and Komodo Edit are recommended. A large set of convenience macros have been written for Komodo (see [Appendix C](#appendix_c) ). See [Section 4](#section_4) to try on-c-e without installation.

The IEP interface consists of four panes that contain various tools arranged by the user. The upper left pane (Figure 3) is an interactive **IPython shell** and the lower left pane is a code editor. When code is run the results are executed and displayed in the shell where they can be interactively explored. As code is typed in the shell or editor, the upper right pane automatically displays code help if the help tab is selected. Other tools display workspace variables and code structure. The lower right pane is a file browser.

1. Download **onceutf.py** and example templates from <http://on-c-e.info>. Copy **onceutf.py** to **pyzo/Lib/site-packages/** and the examples to any folder with write privileges.
2. Open the calc input and calc by double clicking on the files in the file browser. Edit the model and re-run it by scrolling to the run command entered in the shell. Variable values are stored in the shell. Equations can be copied from the calc to the shell for interactive evaluation.
3. Open the calc input and calc by double clicking on the files in the file browser. Edit the model and re-run it by scrolling to the run command entered in the shell. Variable values are stored in the shell. Equations can be copied from the calc to the shell for interactive evaluation.
4. Run **conda** or **pip** from the shell or Scripts folder to install additional packages without affecting other Python installations. The basic Pyzo works with **onceutf.**



**Figure 12. IEP screenshot**

Appendix B - Add a template to calc-db

This section describes the simplest approach to adding a template to the database **calc-db**. The purpose of the exercise is to gain an initial familiarity with the process.

We will use the template **0101.basic.txt** and modify it to make a new template. Everything is done in a browser – no installation is needed.

Step 1. Go to <http://on-c-e.org>. Click on [**[Download Model Templates]**](http://structurelabs.knackhq.com/oncedb#home/)  at the top of the page.

Step 2. Scroll down to **search once-db**, click on the **description** dropdown box and select **contains**. Enter the search term ‘**basic template’**. Click on the **Search** button. One or more results should appear at the bottom of the page. If you search only on ‘**example**’ you will see a number of templates.

Step 3. Look for the template name **0101.basic.txt** and the associated calc names **cal0101.basic.txt** and **cal0101.basic.pdf.** Click on the links that begin with **cal** to view the two different calc formats in a browser window.

Step 4. Select the window for **cal0101.basic.txt.** Press the keys CTRL-A or select all of the template text by dragging the mouse. Press CTRL-C or right click in the browser window and select copy to copy the text to the clipboard.

Step 5. Click on  [**[Upload Model Templates]**](http://structurelabs.knackhq.com/oncedb#upload-model/)  at the top of the page. Scroll down to the box labeled **Template Text** and enlarge it by left clicking on the handle in the lower right corner and dragging it. Click in the box and press CTRL-V or right click and select paste. You should now have a copy of the template in the box.

🖳

🖳

##### **Edit Template**

Step 6. Edit the text, terms, and equations. Leave the tags and vertical bars in place and edit the text between them. Section 6 provides an overview of tags and Appendix B describes tag syntax.

When you finish editing the template, type a brief template summary in the Template Summary Description box above. Include the phrase ‘learning template’ in the description. Delete all of the words ‘example’ from the template and description (so it doesn’t show up in a search). The learning templates will be removed from the database every few days.

##### **Upload Template**

Step 7. Click on the Submit button. You will receive a confirmation message that the template was submitted.

**That’s it! Thanks for learning how to contribute.**

(note: typically the template file will be prepared on your computer and upload

Appendix C - Write a calc in the cloud

This section describes a basic approach to running an example template when a Python platform is not available. It uses a web platform that does not require program installation.

**on-c-e** is designed to run on workstations with Python installed, but it can also run in the cloud on Python web platforms like **Wakari** and **PythonAnywhere.** They provide an in-browser editor and shell terminal interface. **Wakari** and **PythonAnywhere** are discussed in the manual but other Python web platforms can be used. Appendix E provides additional details for web execution. Appendix D discusses installation of the program on a computer or mobile device.

For this exercise a simple example model is run on a Python web platform using the program onceutfnnn.py. You can preview and download other models at <http://on-c-e.org>. Printed example models and calc output are provided in Appendix I. onceutf.py is a single file subset of **on-c-e** that does not include project options, PDF calcs, or unit overrides. It was written to simplify program testing and implementation on web and mobile platforms.

🖳

##### **Run template**

1. Sign up for a free Wakari account at <http://continuum.io/>   
    or   
   PythonAnywhere at <https://www.pythonanywhere.com/‎>
2. Download onceutfnnn.py and **0101.basic.txt** or **0101.basic.txt** from   
   <http://on-c-e.info>
3. Upload the program and template files to a folder on your Wakari or PythonAnywhere account. Choose a bash shell terminal from the drop-down list and open it (Appendix E). At the shell command line check that both files are in the folder you selected (type ***ls*** or ***dir***). Enter the command

python onceutfnnn.py **0101.basic.txt** -e

where *nnn*  is the program version number (i.e. 040).

Processing a calc input will write several files to the directory (Table 2) and echo the log file and calc output, **cal0101.basic.txt**, to the screen. You can open the calc outptu using the browser editor provided by the web platform. The switches following the model name echo the calc to the console **(–e)** or browser **(–b)**. The browser option works when run on local computer.

**Congratulations on running your first model!**

(note: refer to Appendix D for **on-c-e** installation instructions on a computer

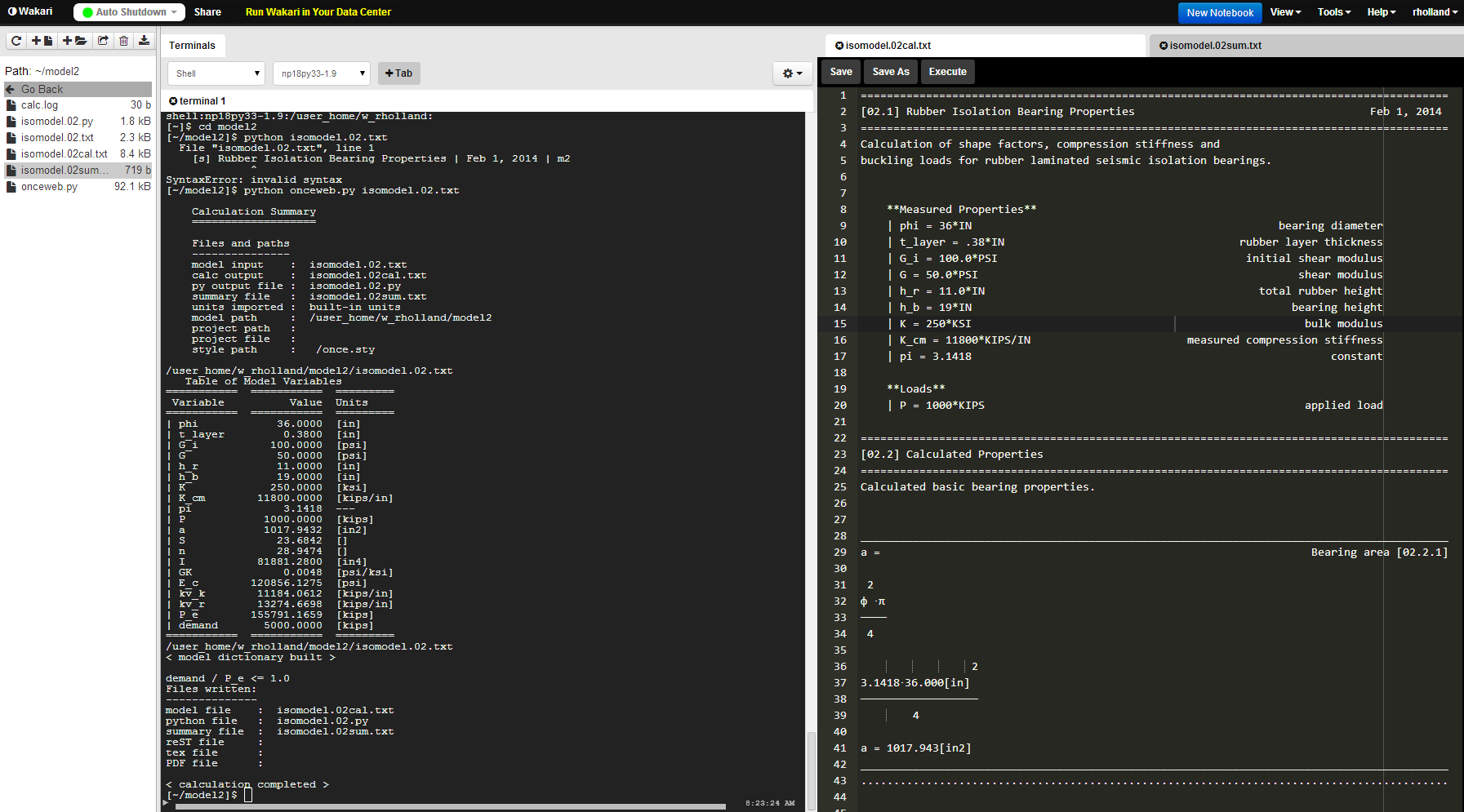
**onceutf.py** runs on Python web platforms. **Wakari** and **PythonAnywhere** are discussed.

**Wakari** is a Python-Linux web platform that runs in the browser. Upload **onceutfnnn.py** and **mode.01.txt** to your Wakari account. Open a **bash shell** window (not Ipython) and type

***python onceutfnnn.py 0101.simple.txt –c***

at the command prompt, where nnn is the version number (i.e. 040). The program will output a model summary followed by the calc output (the –c option echoes the output to stdout; use –b to echo output to the browser when using the program locally). Use the browser file editor to review and modify the files.

C:\Users\rhh\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\21.jpg



This is a calc opened in a Wakari editor. Double click the file name in the file dashboard to the left.

This is a calc log in the terminal window after running the model from the terminal window command line.

**Use Anaconda 1.8 or 1.9 for shell**

**Figure 13. Wakari – terminal and editor in split windows**

**PythonAnywhere** is a Python-Linux web platform that runs in the browser. Upload **onceutfnnn.py** and **mode.001.txt** to your **PythonAnywhere** account. Open a **bash shell** window (not Ipython or Python) and type

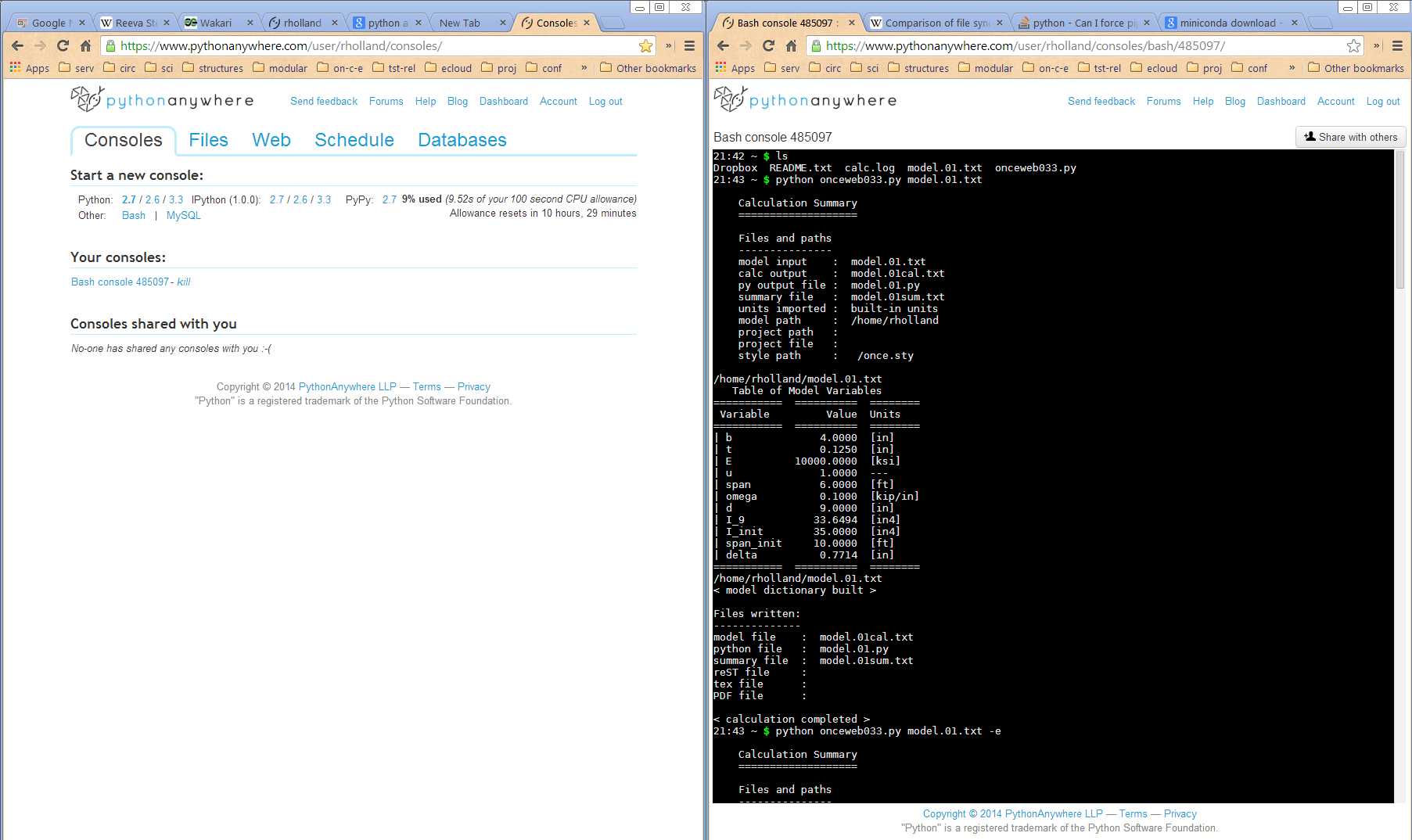
***python onceutfnnn.py 0101.simple.txt –c***

at the command prompt, where nnn is the version number (i.e. 040). The program will output a model summary followed by the calc output (the –c option echoes the output to stdout). Edit files in the file editor.

This is the file dashboard. Click the icon adjacent to the model or calc file name to open in an editor.

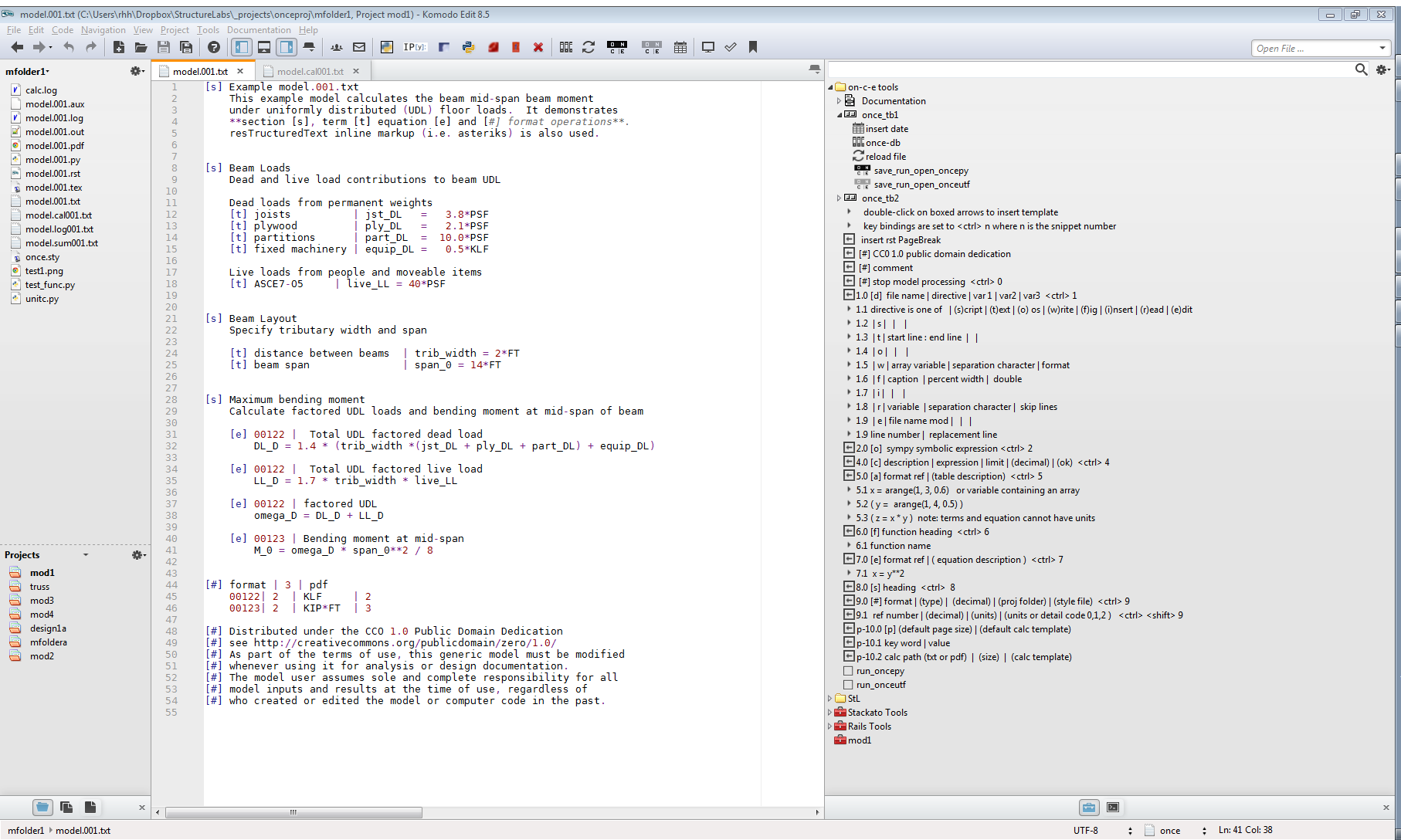
This is a calc log in the terminal window after running the model from the terminal window command line.

C:\Users\rhh\AppData\Roaming\PixelMetrics\CaptureWiz\Temp\8.jpg



**Figure 14.** **PythonAnywhere – side by side browser windows**

# Appendix D – Write calcs using Komodo

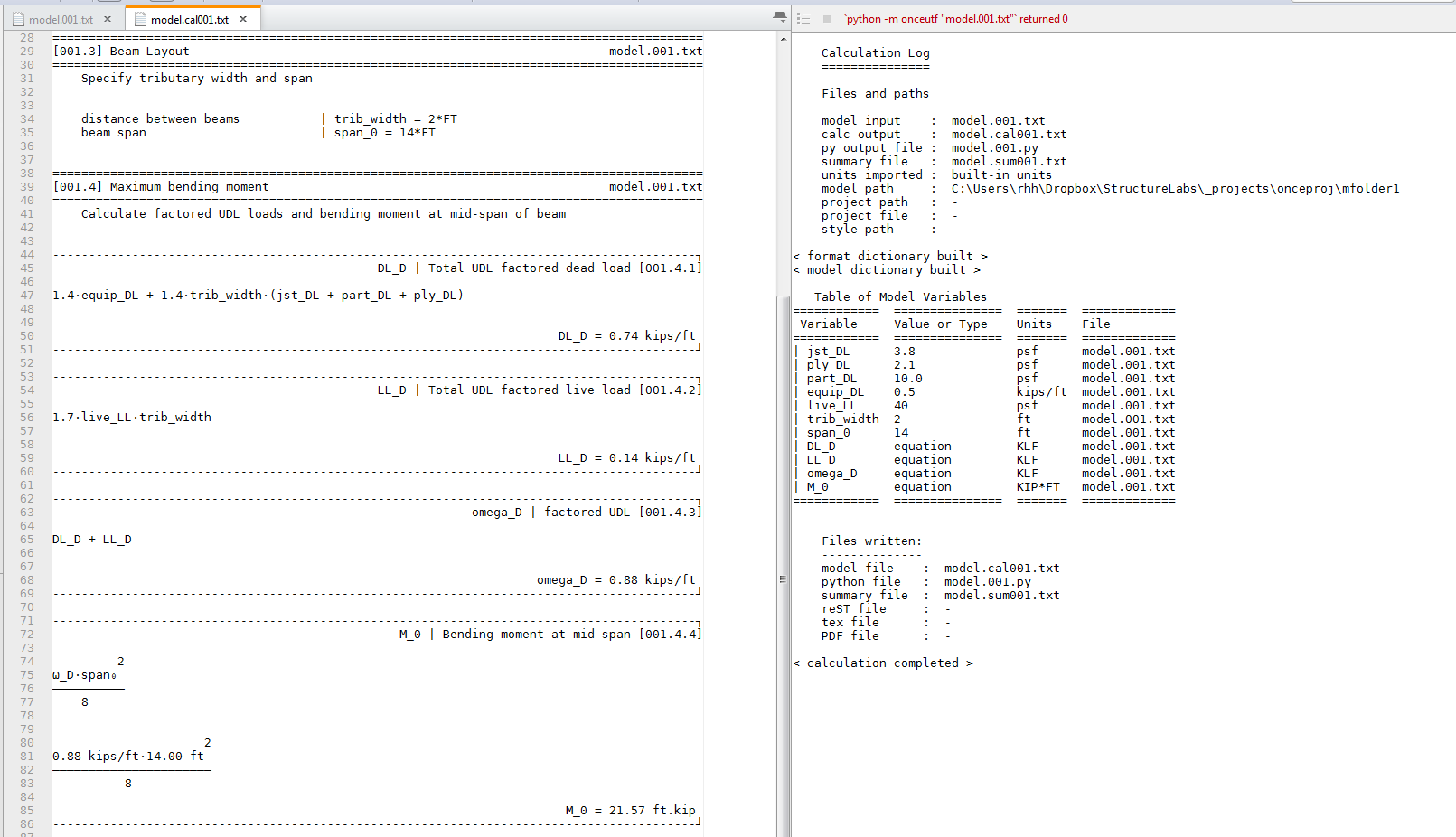


file manager

model input

on-c-e toolbar

operation prompts



UTF-8 calc output

echo of log output

**Figure 15. Komodo interface**

Model files can be prepared using any text editor however Komodo Edit or Komodo IDE provide a significant number of useful capabilities and a number of **on-c-e** convenience macros have been written. Toolbar and menus, key binding, syntax coloring, operation tag templates and processing macros are provided to improve readability and workflow.

**Install on-c-e utilities and fonts in Komodo and set preferences**

1. Download and install DejaVu fonts from <http://dejavu-fonts.org/wiki/Main_Page>

**[1] open   
once-db**

**[13] run model   
in oncepy**

**[15] insert date**

[**14] run model   
in onceutf**

**[12]   
reload   
file**

**[5] close   
all files**

**[4] open PDF calc   
 from UTF calc**

**[9] find in all files**



**[3] open  
file via os**

**[2] clear output pane**

**[6]   
add [3] to context menu**

**[7]   
add [5] to context menu**

**[10]   
open a template**

**[11] execute remote file on remote oncepy**

**[8] run model in IPython**

**The on-c-e toolbar (once\_tb1) operates on the file that has editor focus**Drag and drop PDF files into a viewer from the folder pane, right click for context option or set and use macro button [4]. A viewer that does not lock the file (i.e. a browser) is recommended.

1. Download and unzip **oncetoolsnnn.zip**
2. **Install the on-c-e tools folder** by right clicking in tool box pane – import folder – select **on-c-e tools** folder from the unzipped file. Tools include a **tool bar**, **documentation menu** and **tags** that can be inserted into models by double clicking.
3. Close and reopen Komodo to complete installation. Edit a tool by right clicking on it in the tool pane and selecting Properties. Pane arrangement is adjustable.
4. **Install the syntax coloring plugin** **once-0.3.1-ko.xpi** using the menu - **Tools – Add On.** Select the xpi file from the unzipped **oncetoolsnnn** folder. To set **once** syntax coloring for a file use the context Properties and Settings when the file is open in the editor and select **once** as the Language in the File Preferences settings.
5. After running a model using buttons [13] or [14], or the shortcuts <Ctrl-Spacebar> or <Ctrl-Shift-Spacebar>, the UTF calc gets focus in the editor pane. Set the command dialog open\_pdf\_calc\_from\_utf\_calc to the preferred PDF viewer to open the associated PDF from this tab using CTRL-P or the tool bar button.
6. The suggested model line length is 90 characters using mono-space 9pt font. Indents should be spaces, not tab characters. UTF calcs are formatted to a 90 character width. Set line background color and search highlighting in **Edit–Preferences-Fonts and Colors–Colors** or **Indicators.**
7. Copy the file **once\_template.txt** from the **once tools** folder to the **My Templates** folder in the Komodo user directory
8. **Recommended pane arrangement and keyboard bindings**

Set pane locations by right clicking on tabs or icons.

Places (Projects and files) - left pane

Editor - middle pane

Command Output and Toolbox - right pane

1. **Copy once\_keybindings.kkf to your settings directory and set in Preferences.**

For a list of active bindings select from menu: **Help – List Key Binding**

If context menu macros do not load when the program starts double-click on the following macros in the Toolbox once\_macros folder to activate them:

Z1context\_comments adds block comments to context menu (Editor)

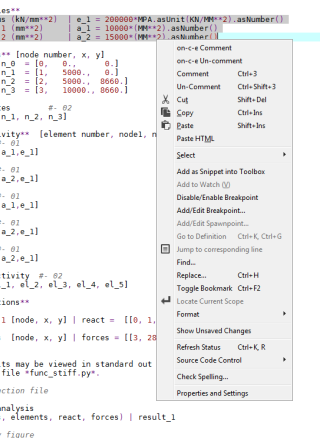
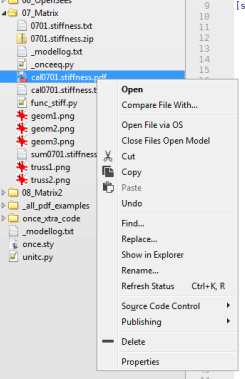
Z2context\_osfile open file using OS default association (Places)

Z3context\_modelfile close open files and open selected file (Places)

Z4context\_terminal open terminal in selected folder (Places)

Z5context\_oncecomments adds block comments to context menu (Editor)

The once-specific context menu entries are shown below.



**Model operations**

CTRL-0 insert break #stop

CTRL-1 section [s]

CTRL-2 symbolic [y] |

CTRL-3 term [t] |

CTRL-4 check [c] | |

| |

CTRL-5 array [a] #-

CTRL-6 function [f]

|

CTRL-7 equation [e] #-

CTRL-8 format #- formateq | 3, 3 | 1.0

#- 01 | 2, 2 | | 3

CTRL-9 file #- fileop

#- 01 | s | file | | |

CTRL-SHIFT-Z insert license public domain license

CTRL-[ block comment z0\_oncecomment (macro)   
CTRL-] block un-comment z0\_onceuncomment (macro)

**IDE navigation**

CTRL-SHIFT-SPACE run onceutf ***save\_run\_open\_onceutf*** macro

CTRL-SPACE run oncepy ***save\_run\_open\_oncepy*** macro

CTRL-P open PDF from UTF calc ***open\_pdf\_calc\_from\_utf*** macro

CTRL-SHIFT-A save file as ***save\_as*** macro

CTRL-SHIFT-B toggle bottom pane **on-c-e** key binding

CTRL-SHIFT-C close all ***close\_all\_files*** macro

CTRL-SHIFT-D focus directory pane ***focus\_directory*** macro

CTRL-SHIFT-E focus edit pane Komodo default

CTRL-SHIFT-F find text in all ***find\_all\_selected*** macro

CTRL-SHIFT-I run onceipy ***copy\_equa\_run\_IPython*** macro

CTRL-SHIFT-K search toolbox Komodo default

CTRL-SHIFT-L toggle left pane Komodo default

CTRL-SHIFT-N new file from template ***new\_file\_template*** macro

CTRL-SHIFT-O focus output pane ***focus\_output*** macro

CTRL-SHIFT-Q set file preference ***set\_file\_pref*** macro

CTRL-SHIFT-R toggle right pane Komodo default

CTRL-TAB next editor tab Komodo default

CTRL-SHIFT-T toggle toolbox display ***show\_toolbox*** macro

TAB exits search

1. Navigate between correlated locations in the model and calc: Select text in a model or calc. Select **find\_all\_selected** from the toolbar < Ctrl-Shift-F>. Step through results with CTRL-TAB and F3 or select from list with mouse. Line location in file is maintained between runs.

🖳

# **C:\Users\rhh\Dropbox\StructureLabs\admin\logos\icons\ipy_icon.png**Appendix E – Evaluate calc equations interactively

1. **Komodo IDE** (commercial version): Cut and paste the entire \_**onceeq.py** file into the interactive shell for access to equations and terms.
2. **IPython Notebook**: <http://ipython.org/notebook.html>   
   Cut and paste the entire \_**onceeq.py** file into a notebook cell for interactive access to equations and terms.
3. **IEP (**see Section 2). Cut and paste equations and terms from the **\_onceeq.py** file into the shell for editing. If more than 1 equation at a time is pasted it is treated as a single block.

**4. IPython Terminal** and **Qt shell**: <http://ipython.org/>

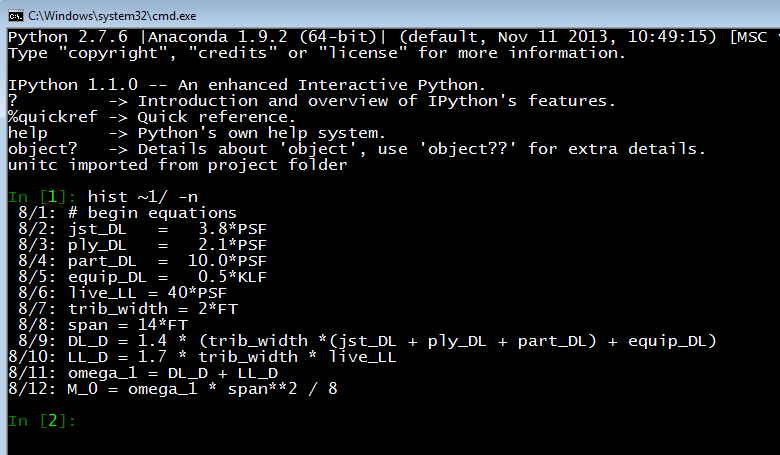
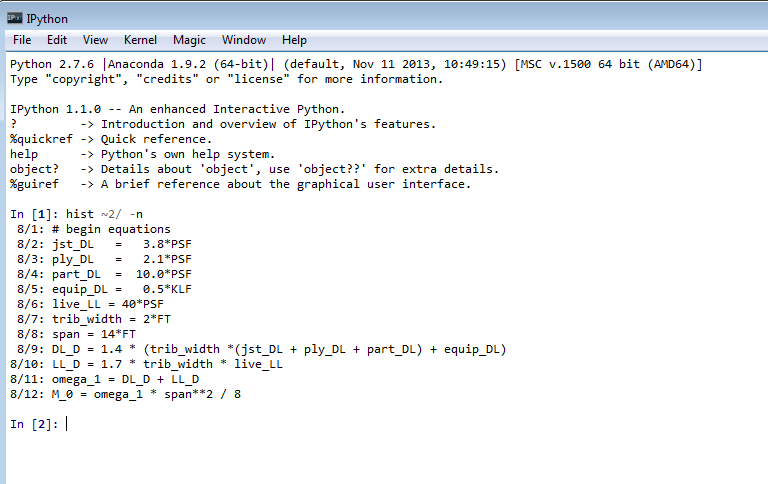
Method 1. Open IPython and run %autoindent (toggle off). Cut and paste content of **\_onceeq.py** into IPython.

Method 2. Copy **onceipy.py** to the Python /lib/site-packages directory and run it from the model folder. Run the script **onceipy.py** to update the IPython **history.sqlilte** database with the equations in **\_onceeq.py**. Next, run **\_onceeq.py** in IPython to set up the environment. Both steps are executed by the **IPy** macro button (Ctrl-Shift-I) on the Komodo **on-c-e** toolbar ([Appendix C](#appendix_c)). Run the macro after running the model and when the model or calc has focus. Modify the command "**run\_onceipy**" (at the end of the tool list in the Komodo toolbox) by adding the location of the history.sqlite directory to the command line entry (without brackets), i.e.

python -m onceipy.py [C:\Users\rhh\.ipython\profile\_default]

**onceipy** reads model equations from the \_**equations.py** file previously generated by **onceutf** or **oncepy**. The equations and terms are added to the **history.sqlite** database and assigned the next session number. A backup copy of the file is made before the new equation records are added. If the history.sqlite file becomes too large or corrupted it may be deleted and then reinitialized simply by running IPython from the command line. After reinitializing, exit IPython before adding new models to the history database. Equations and terms are available in the shell through command history scrolling with up and down arrows (type the first few letters to filter), and may also be accessed with search (Ctrl-r) and the **%hist *ref*****–n** command

* where ***ref*** is of the form **s/e** **–n** (-n numbers the entries)
* **s** is the session number, **e** is the range of entries, use **s/** for all
* **s** may also be **~s** which is the relative session number counting back,   
  i.e. **~1/ -n** lists and numbers the most recently added **on-c-e** model.
* recall a history line with **rep *num***  where ***num***is the history reference number
* array equations are written to the model history with quotes to avoid unit conflicts in initial processing.

****

**Figure 16. Screenshots of model equations in IPython shells**

# Appendix F – Generate PDF calcs on workstations

**Run a local folder copy of onceutf.py on Linux, Windows, Mac:**

Install a scientific Python distribution.

Install UTF font symbols**:** [**http://dejavufonts.org/wiki/Main\_Page**](http://dejavufonts.org/wiki/Main_Page)

Copy **onceutf.py** and the model file into the same folder, open a console window in the model or project folder and type:

**For links see**

**Section 9**

***python onceutf.py ffdd.name.txt (-t or –b) (-noclean) (-nn)***

where **ffcc.name.txt** is the file name of the input model; **-t** or **–b** are options to echo calculation results to the terminal or browser; **-nn** is the two digit line length (default 80) and **–noclean** retains the auxiliary PDF files.

To open a terminal window in in Windows 7 or 8, navigate to the folder with the model, hold the shift key, right click in the folder and click on 'open command window here'.

Change the browser encoding settings if needed:

* Chrome – type **chrome:settings/fonts** in url bar - scroll to the bottom of the dialog box to make the change.
* Firefox - options - content - advanced - UTF-8
* Internet Explorer - right click in window - encoding - UTF-8

**Run a ‘site-packages’ copy of onceutf on Linux, Windows, Mac:**

Install a scientific Python distribution. Copy the **onceutfnnn.py** file into the Python/Lib/site-packages/ folder and rename to **onceutf.py.** Run from any directory with the command

***python –m onceutf m.modfile.txt***

**Run the oncepy environment on Linux, Windows, Mac:**

1. Install Anaconda 1.9.2
2. Install unum, tabulate and PIL modules (using pip install)
3. Install TexLive
4. Install DejaVu Fonts and set to Mono in IDE
5. Install Komodo Edit or IDE
6. Install **on-c-e** tools for Komodo
7. Unzip and copy the **oncepy** folder into *Python/Lib/site-packages/*
8. Run from any model or project folder containing the model file:

***python –m oncepy nnnn.modfile.txt***

# Appendix G – Write UTF calcs on Android and iOS

**iOS**

**onceutf** runs in the iOS **Pythonista** **App** environment which includes numpy, sympy and matplotlib libraries. Copy the **onceutf.py** file to the site-packages folder and a model file into a (division) folder.

Without Dropbox integration files are moved by creating new empty files in Pythonista and using copy and paste. A script for downloading Dropbox files to Pythonista is at:

<https://omz-forums.appspot.com/pythonista/post/4995757044137984>).

Open the **onceutf.py** program in Pythonista and create an ‘action’ (wrench menu) for it, leaving the argument blank. Open a model or template. Run it by invoking the action. The calc will be echoed to the screen. File management, searching, UTF-8 fonts and word wrap functions are available in **Textastic** or **Notesy Apps.** The calc can be opened using an action and the following script where program is notsey or textastic.

**open\_calc.py script (**<http://on-c-e.us>)

"""open Pythonista calc in app

1. Copy this script to Pythonista and set it as an action.

2. Set the fixed width font to DejaVu Sans Mono.

"""

import editor

import webbrowser

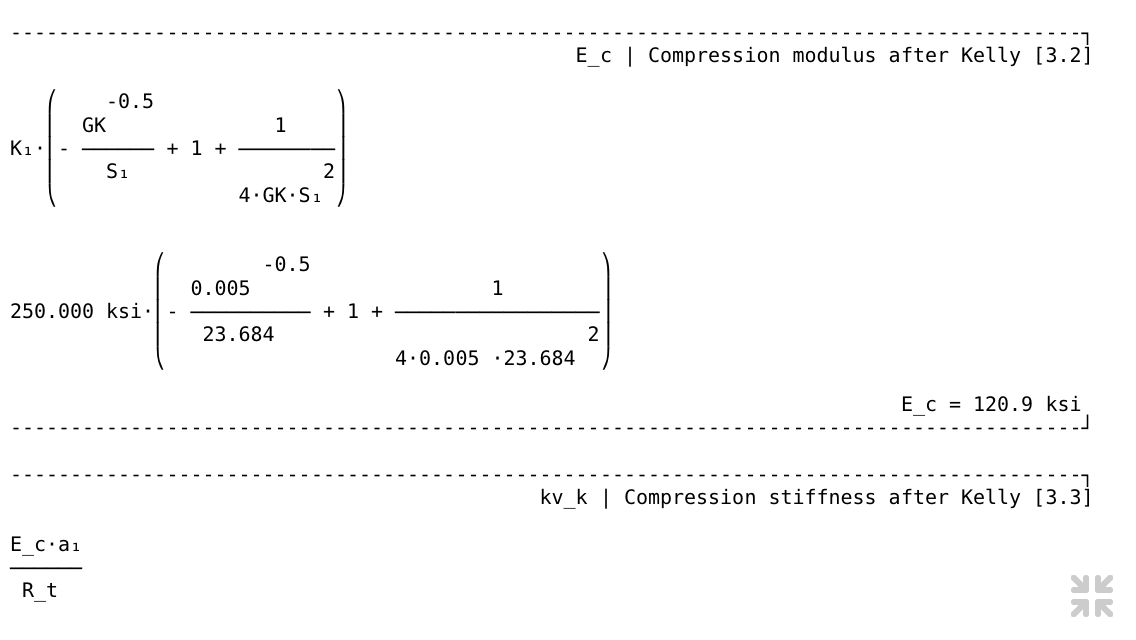
import clipboard

base = 'program://x-callback-url/append?name=calc\_tmp.txt'

text = editor.get\_text()

url = clipboard.set(text)

webbrowser.open(base)



**Figure 17. Screenshot of onceutf calc on iPhone**

**Android**

**onceutf** runs in the **QPython** **App** (Python 2) environment on Android. Copy the compiled numpy and sympy libraries into the site-packages folder at

/mnt/sdcard/com.hipipal.qpyplus/lib/python2.7/site-packages.

The compiled libraries are available through QPython sites and also at <http://on-c-e.us>. Matplotlib libraries are not yet available.

The following file arrangement and script makes it convenient to iterate and review a model. It uses the **DroidEdit App** to edit the model and view the calc. The **Hackers Keyboard App** is efficient for writing models.

**once.py script (**<http://on-c-e.us>)

#qpy:console

"""

once.py script runs a onceutf model on Android using QPython

1. Copy onceutf.py to the ‘scripts’ folder (see prog path below)

2. Create a division folder for the models (see model path below)

3. Copy the model and this script into the division folder

4. Open this script in QEdit (which is part of QPython)

5. Modify folder and model names below, as needed, and save.

6. Edit models and review calcs in DroidEdit (or similar editor).

7. Set the editor font to DejaVu Sans Mono for calc math symbols.

8. Run this script in QEdit to execute program.

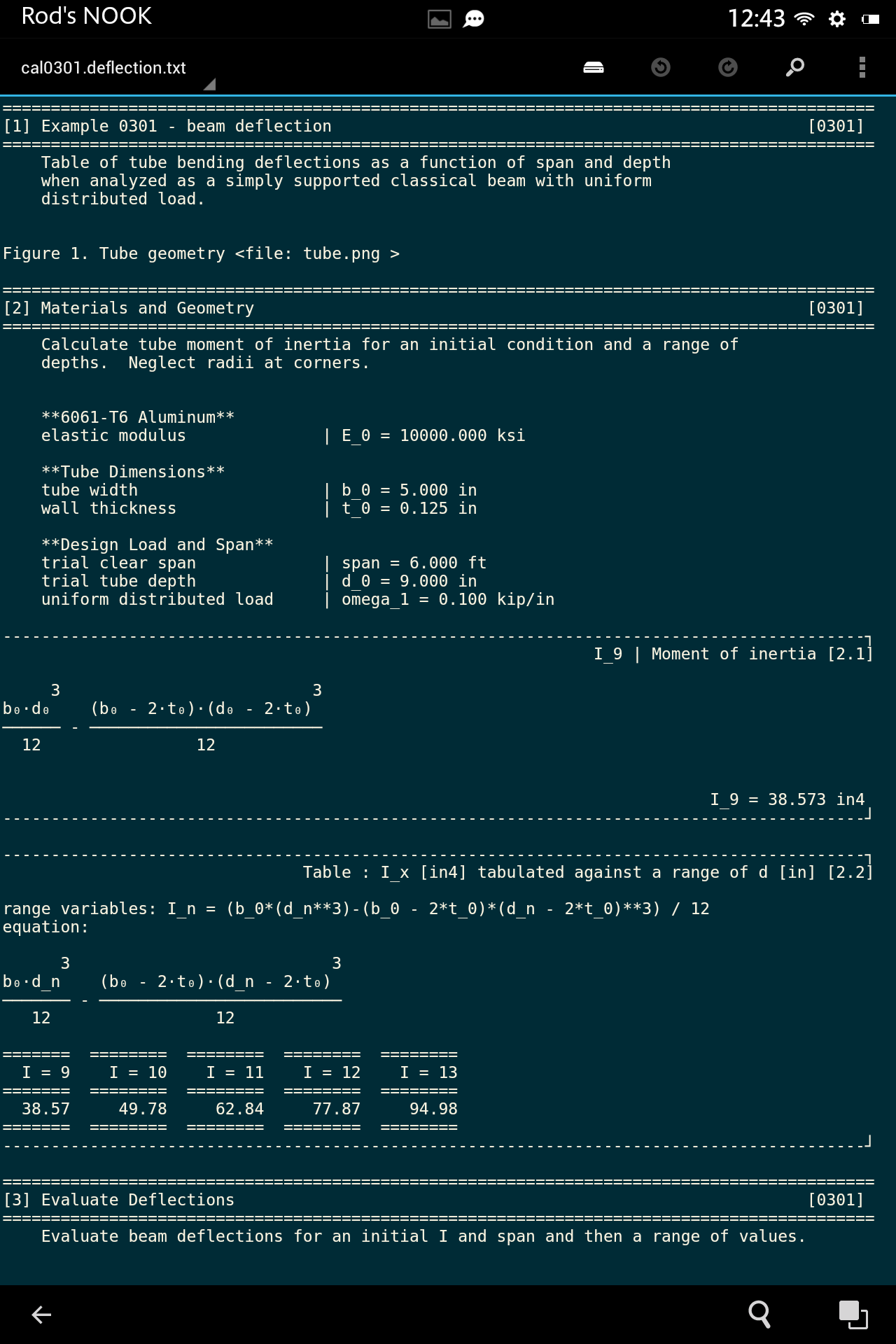
"""

import os, sys

prog ='/mnt/sdcard/com.hipipal.qpyplus/scripts/onceutf.py'

model ='/mnt/sdcard/com.hipipal.qpyplus/models/0301.deflection.txt'

os.system(sys.executable + “ “ + prog + “ “ + model)



**Figure 18. Screenshot of onceutf calc on Android tablet in DroidEdit**

# Appendix H – Minimum Programs and Libraries

Note: If the standard scientific distribution does not include a library, use the pip install method or in the case of Anaconda use conda management tool. Other scientific libraries including SciPy, pandas, rpy etc. may be installed by the distribution or installed separately and will be available to **on-c-e** through functions and script operations (see file operations, Appendix B).

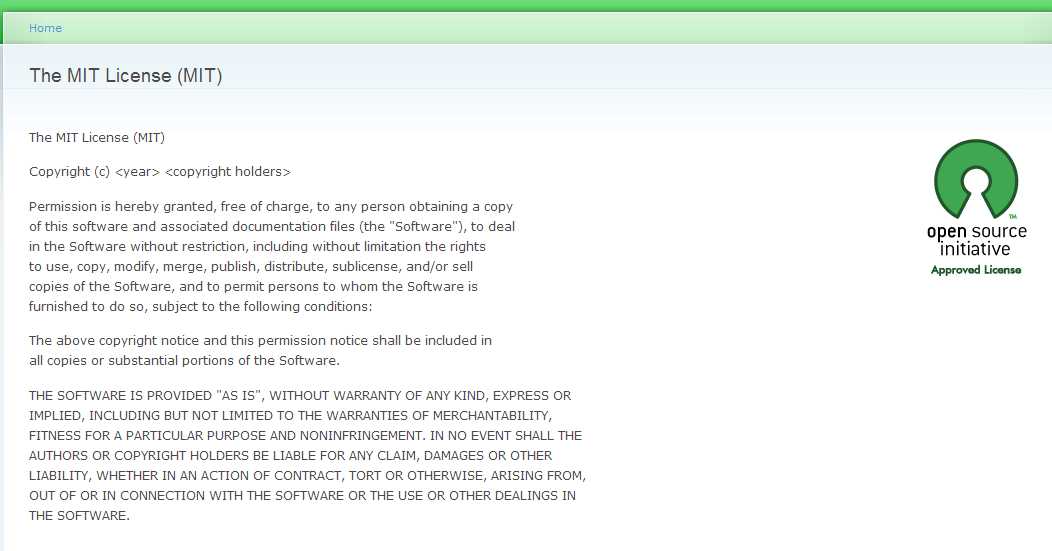
|  |  |  |  |
| --- | --- | --- | --- |
| **Program** | **Python 2** | **Python 3** | **Notes** |
|  |  |  |  |
| **onceutf** | **0.4.6** | **0.4.6** |  |
| **Python** | 2.7.x and above | 3.4 and above | Python Platform |
| **Numpy** | 1.7.x and above | 1.8 and above | Python Library |
| **SymPy** | 0.7.x and above | 0.7.5 and above | Python Library |
| **Matplotlib** | 1.2.x and above | 1.3.1 and above | Python Library |
| **DejaVu fonts**  **Unum** and **tabulate** are built-in | 2.3.4 and above | 2.3.4 and above | UTF math fonts (external program) |
|  |  |  |  |
| **oncepy** in addition to above | **0.4.6** | **0.4.6** |  |
| **Unum** | 4.1.x and above | 4.1.x and above | pip install |
| **Tabulate** | 0.7.x and above | 0.7.x and above | pip install |
|  |  |  |  |
| **oncepy with PDF** in addition to above | **0.4.6** | **0.4.6** |  |
|  |  |  |  |
| **PyPDF2** | 1.1x and above | 1.1x and above | pip install |
| **PIL** | 1.1.7 and above |  |  |
| **Pillow** |  | 2.5.1 and above |  |
| **TeX Live** | 2013 and above | 2013 and above | Network install (external program) |
|  |  |  |  |
|  |  |  |  |

**Table 5. Minimum required programs and libraries**

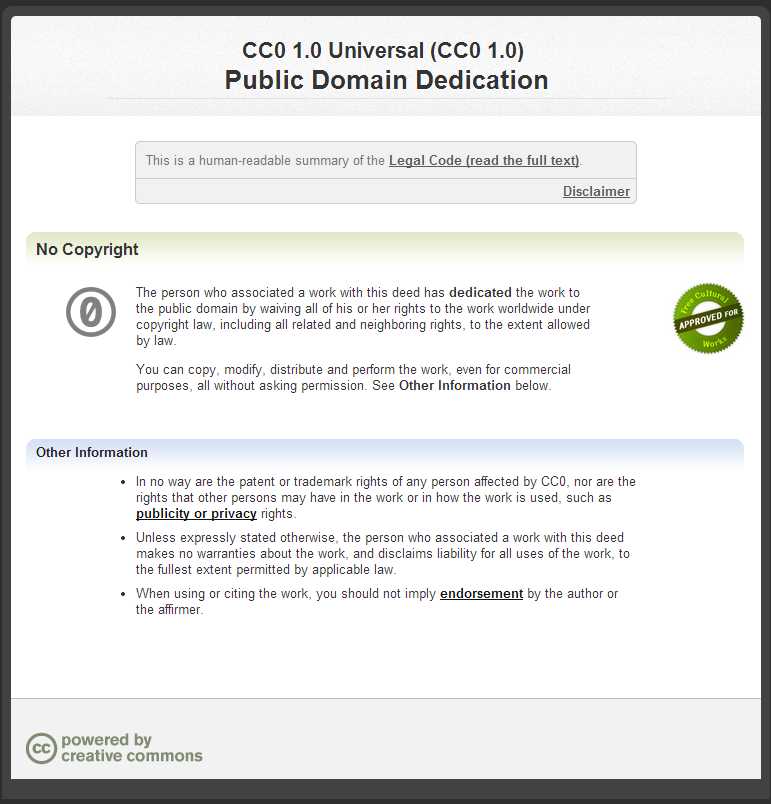
🗁

# Appendix I – Licenses

**oncepy, onceutf** and **on-c-e Komodo tools** are distributed under the **MIT** or compatible license. Copyright is retained by the code author.



**Figure 19.** [**http://opensource.org/licenses/MIT**](http://opensource.org/licenses/MIT)



**Figure 20. http://creativecommons.org/publicdomain/zero/1.0/**

Template files uploaded to **once-db** are made available to others under the CCO 1.0 Public Domain Dedication. A **Komodo** snippet is provided for inserting license text in a template file:

**# This file contains a on-c-e public domain template (the template).**

**# The template is distributed under the CCO 1.0 Public Domain Dedication**

**# at http://creativecommons.org/publicdomain/zero/1.0/**

**# The template is not a structural design calculation.**

**# The template user assumes sole responsibility for all inputs and results.**

If the license is in the template the following license is added to the calc.

**This document (the calc) is generated from a on-c-e public domain template.**

**The calc is licensed under the CCO 1.0 Public Domain Dedication**

**at http://creativecommons.org/publicdomain/zero/1.0/**

**The calc is not a structural design calculation. The calc user**

**assumes sole responsibility for all inputs and results.**

👓

# Appendix J – Security and Namespaces

**on-c-e** is designed to be a flexible and efficient tool for structural analysis and design. It is also designed to facilitate program verification by keeping the source code simple and transparent. **on-c-e includes** about 2000 lines of source code organized in less than a dozen modules and built on the large scientific library collection. Programming tradeoffs occur in pursuing these objectives.

The programs use Python **exec** and **eval** statements because they are most straightforward way to process the model operations. It is possible to send intentionally malicious input to **exec** and **eval**. This risk is controlled or eliminated with relatively little effort on the part of the user. Sensible model file input is obvious and constrained. It is under the control of engineers and designers who are running the models and can recognize appropriate input, in the form of structural analysis equations, arrays and functions. the engineers using the tool. Because unsafe code in the model would appear very different from this well understood structural text it is straightforward to spot and delete suspicious or problematic model code before running. In the future additional security checks may be added to the program but the only way to ensure proper operation of the program is to check the input.

The programs also import **numpy**, **sympy** and other libraries in their entirety into the model namespace. This approach improves readability and streamlines equation input but can introduce variable name collisions. Variable naming rules have been developed and more will be added as experience is gained and the program matures.

In summary, to make **on-c-e** more readable, useful and flexible, the design philosophy chosen emphasizes code simplicity and transparency and relies on the experience of the user to recognize appropriate input. Verbose and restrictive input policies were avoided.

# Appendix K – Examples

🖳

This appendix includes printed PDF calcs, UTF calcs and model files, grouped in that order, for each example. The latest files including the IPython files, function files, summary files and log files can be downloaded from **once-db** for each example at <http://on-c-e.org>.

**00\_Basic:** 0001.basic.txt **Demonstrates:**Sections, terms, equations, reST, tables

**01\_Basic\_latex:** 0101.base\_latex.txt  
Sections, terms, equations, raw LaTeX, reST, symbolic equations, tables

**02\_IsoBearing:** 0201.bearing.txt   
Figures, complex equations, DC ratio checks

**03\_Deflection:** 0301.deflection.txt  
Figures, arrays, equation expansion options

**04\_Seismic:**

0401.bldg\_info.txt term lists, inline reStructuredText  
0402.seismic.txt built-in math functions  
0403.seismic.txt sub-models, symbolic expressions  
0404.frame.txt sections, array  
0405.brace.txt DC checks

**05\_Dynamics:** 0501.eigenvector.txt  
External functions, plotting, linear algebra library.

**06\_OpenSees:** 0601.truss.txt  
Read, edit, file operations for external programs and data

**07\_Matrix:** 0701.stiffness.txt  
External functions, plotting, linear algebra library

# Index

**0101.template.txt**, 9, 51

2D diagrams, 19

3D diagrams, 19

**Android**, 10, 41, 43, 44

**arrays**, 30, 32, 50, 51

ASCII, 14

calc, 8, 9, 12, 14, 15, 33, 34, 35, 36, 38, 45, 46

calculation, 9, 11, 14, 29, 41

**DejaVu**, 16, 38, 47

Dia Diagram Editor, 19

diagrams, 19

**Download**, 9, 16, 38

equations, 29, 32

fonts, 16, 38, 41

Fonts, 16

**format**, 12, 13, 28, 30, 31, 32, 33, 35, 36

**FreeCAD**, 19

functions, 33, 34, 51

IDE, 10, 11, 16

IEP, 5, 7

**install**, 16, 47

interactive analysis, 5, 10, 14

iOS, 41, 42

IPython, 5, 10, 14, 17, 51

key binding, 38

Komodo, 10, 11, 16, 17, 37, 38, 40, 41, 48, 49

**Komodo Edit**, 16

LaTeX, 5, 10, 20, 36

Linux, 10, 11, 41, 45, 46

**MATLAB**, 11

model, 9, 10, 12, 14, 15, 16, 32, 33, 34, 35, 41, 45, 46, 49

**Notesy**, 42

***on-c-e***, 2, 5, 9, 10, 11, 16, 17, 38, 47, 48, 50

**once-db**, 10, 11, 49

**onceutf**, 11, 16, 41, 42, 43, 47, 50

**OpenSees**, 51

OSX, 10, 11

pane, 38, 39, 40

PDF, 13, 15, 47

**project**, 5, 9, 12, 14, 15, 16, 35, 36

Python, 9, 11, 15, 16, 30, 31, 33, 34, 45, 46, 47

**PythonAnywhere**, 9, 45, 46

**Pythonista**, 10, 42

Pyzo, 7, 10, 16

**QPython**, 10, 43

**reST**, 10, 12, 20, 28, 51

**reStructuredText**, 10, 12, 28, 51

**sections**, 12, 29, 51

**SketchUp**, 19

**sqlilte**, 17

support, 16

template, 8, 9, 11, 34, 36

terms, 29

**Textastic**, 42

**Upload**, 9, 16, 45, 46

UTF, 15, 34, 38, 41, 47

UTF-8, 15, 34, 41

**Wakari**, 9, 11, 45

Windows, 10, 11, 41

**© StructureLabs**

15 Blanca Drive

Novato, CA 94947

Phone 415.314.8453 **• rholland@structurelabs**