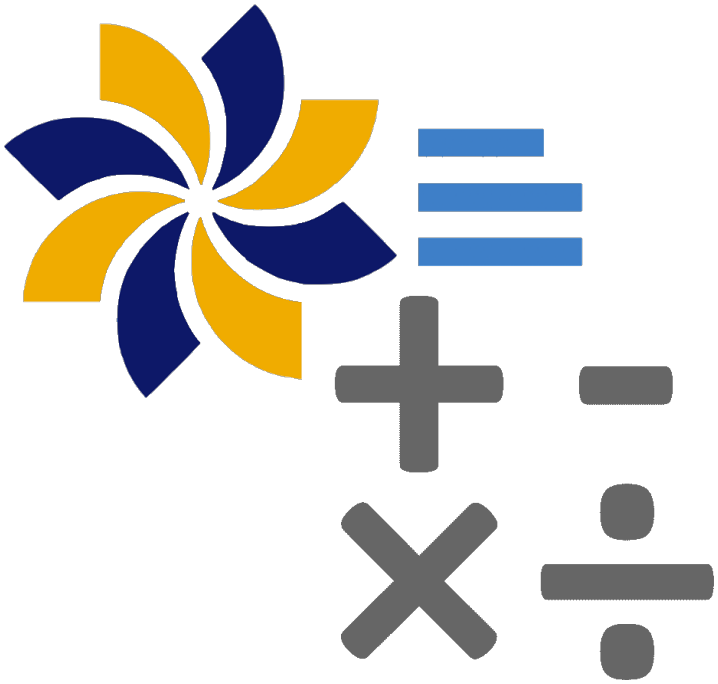
CalcIt's Manual

Version 2.0.0



Zynergy Apps, LLC

Table of Contexts

Table of Figures

Table of Tables

1 Introduction

Welcome to Zynergy Apps' Calculator – The Middle Path of Computing. This app can be used to perform simple calculations like a traditional calculator in addition keeping list of previous calculations for editing purpose to perform "what ifs" computations.

To get started quickly, please review the *Getting Started* and *Quick Tour* sub-sections. This will give you an overview of the app's capabilities within 30 minutes. The rest of this guide is designed to introduce general app capabilities to take you to the next level of calculating numbers of interest!

1.1 Getting Started

On the *Calculator* tab, enter an expression to calculate in the "Enter expression to calculate" field. For example, click in the field, enter `1 + 1,` and press Return to compute the value of `2`.

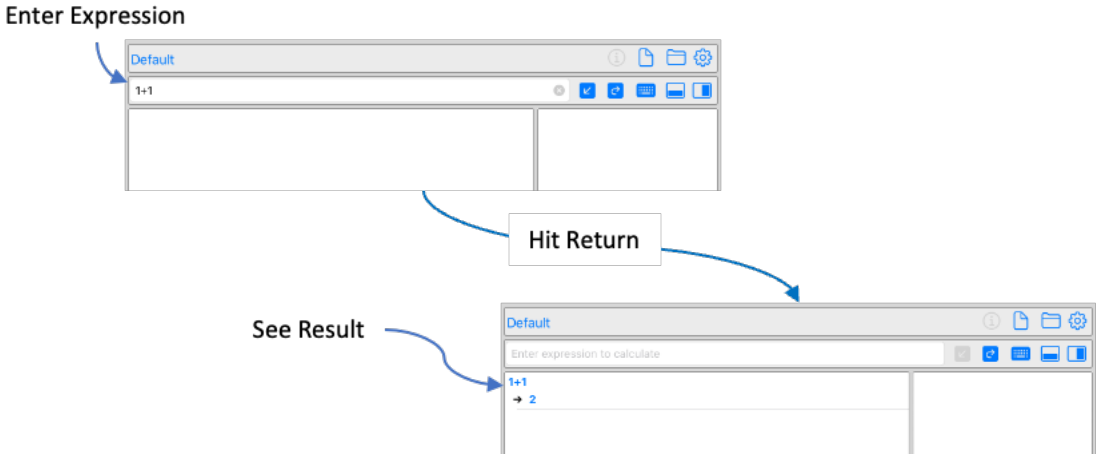


Figure 1 - Calculate 1 + 1

Naturally, you can compute more complicated calculations. For example, say your hourly rate at your place of employment is `$25.48,` and you work `40` hours a week with two weeks of vacation included. What is your annual salary? Enter `25.48 × 40 × 52,` and it will compute `52998.4,` representing your annual salary.

```
25.48 * 40 * 52
→ 52998.4
```

Figure 2 - Calculate Annual Salary

You can beautify the results using the `fmt` function by re-typing in `fmt("$%,7.2f", 25.48 × 40 × 52)` to output `$52,998.40`. Details are in the *Libraries* chapter in the *Convert* sub-section on the `fmt` function formatting options.

```
fmt("$%,7.2f",  25.48 * 40 * 52)
→ $52,998.40
```

Figure 3 - Calculate Annual Salary Alternate

1.2 Quick Tour

Let's review the general parts of the app and its basic functionalities.

⇒ NOTE: Majority of the figure illustrations are from the iPad Mini device.

1.2.1 Menu Bar

On each tab at the top of the screen, a menu bar is displayed to perform various actions by the app.

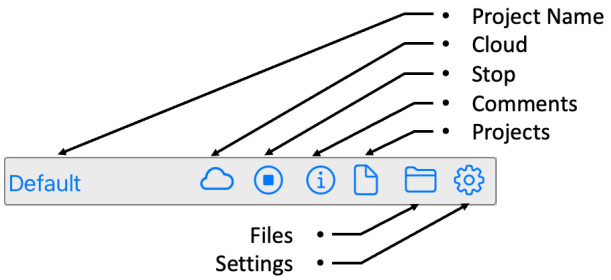


Figure 4 - Menu Bar

- The first part of the menu bar displays the name of the project currently in use.

  ⇒ NOTE: The "Default" project is always available for general-purpose calculations and cannot be renamed or deleted.

- The *Cloud* icon is an indicator the project is on the iCloud. Otherwise, no cloud icon is displayed for local project.

- The *Stop* button allows you to terminate a `loop` command currently running. Normally, this button is not visible unless the `loop` command has been running longer than 3 seconds. Generally, this should never appear since the `loop` command is restricted up to 250 iterations.

- The *Comments* button allows you to add comments to your calculation to remind you of the purpose of the calculation. For example, let's add a comment to the previous calculation in the *Getting Started* section. Also, you can optionality select a color tag to be displayed with the calculation.

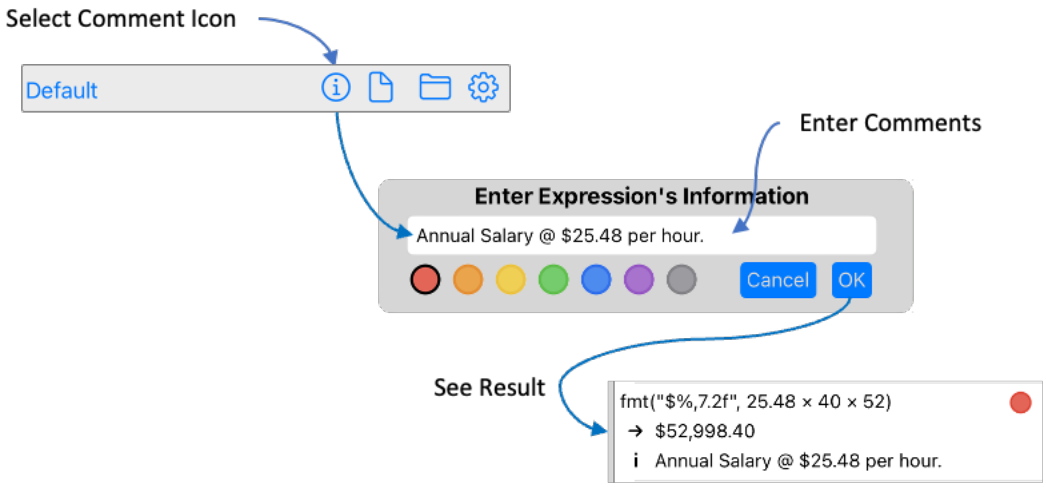


Figure 5 - Adding Calculate Annual Salary Comments

⇒ NOTE: Adding comments and color tag is also applicable to *Calculator*'s Variable column.

- The *Projects* button allows you to select another project. It also includes creating new projects, renaming projects, deleting, and archiving. You can create new projects to organize your working calculations for various computational activities.

- The *Settings* button displays a pop-up menu to configure the app's environment.

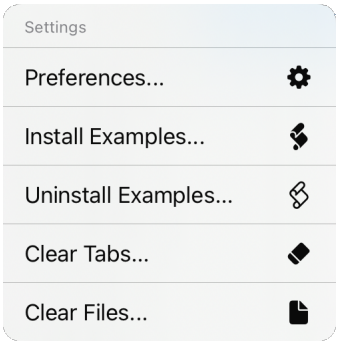


Figure 6 - Settings Pop-up Menu

  - The *Preferences* option displays the app's parameter settings.

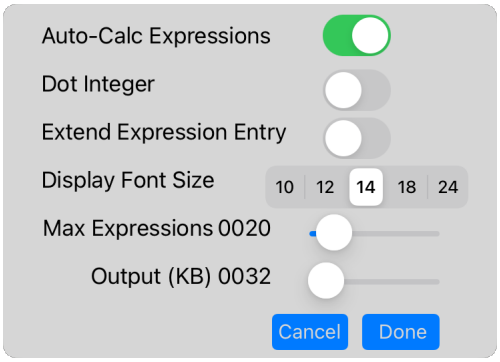


Figure 7 - Settings' Preferences Dialog

    - *Auto-Calc Expressions* enables automatically recalculating your expressions containing variables that have changed. A similar mechanism is found in spreadsheet apps; if a cell is changed, it will trigger an update to other cells. Additional details are in the *Expressions* chapter in the *Variables* section.

    - *Dot Integer* enables integer number results to embed a period character for every third integer digit to allow easy reading of large numbers of millions, billions, and trillions.

    - *Extend Expression Entry* enable extended expression editing with Apple's Pencil in the Calculator's tab.

      ⇒ NOTE: This option is only available for iPad devices.

    - *Display Font's Size* changes the text and field's font size between 10, 12, 14, 18, or 24 point-size.

- The *Max Expressions* slider sets the maximum number of expressions displayed on the *Calculator* tab before rolling off the *Expressions* column display. If the setting is set to zero, no expressions will be removed from the column's display.

- The *Output (KB)* slider sets the maximum amount of memory that can be used to display on the *Outputs* tab before rolling off oldest message output.

    - The purpose of this setting is to avoid excessive messages being output by the scripts.

    - KB means kilobytes, which is 1,024 bytes (i.e., characters) of memory.

- *Install Examples* displays basic expression examples to be installed into the current project for demonstration purposes. Likewise, you can *Uninstall Examples* to remove the installed expressions.

- *Clear Tabs* allows clearing the contents on the *Calculator* and *Outputs* tabs. And there is *All* button to clear everything in the current project.

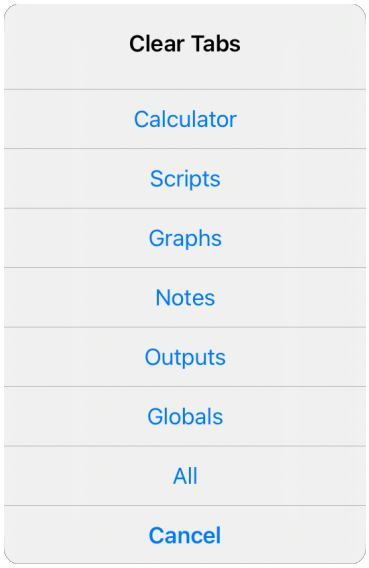


Figure 8 - Settings' Clear Tabs Dialog

1.2.2 Calculator

The *Calculator* tab has a sub-menu bar and three sections. Section **A** is the *Expressions* column, displaying the results of the entered expressions. Section **B** is the *Variables* column, displaying the variables' assignments from the results of the entered expressions. Section **C** is the *Results* area, displaying output of the expressions' results.

You can resize the *Results* area by selecting the little black button on right side of the screen between sections **B** and **C**. This will allow more output viewing of the calculated results.

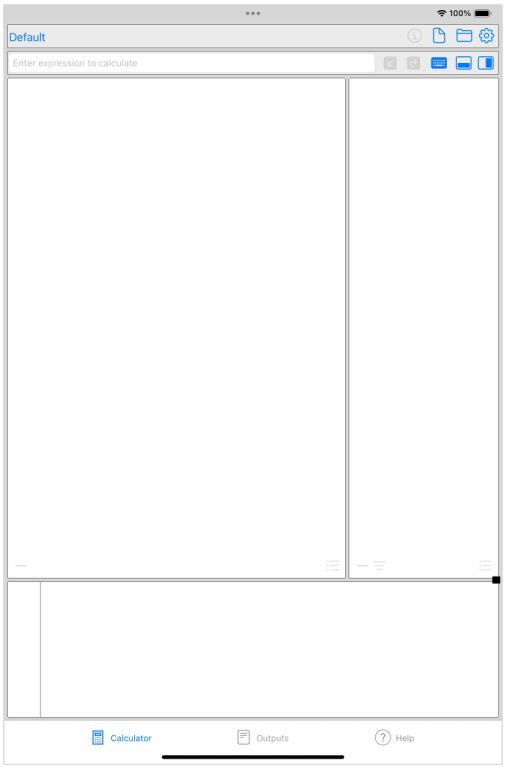


Figure 9 - Calculator Tab

- The first part of the menu bar displays the entry of the expression to calculate in its collection of commands.

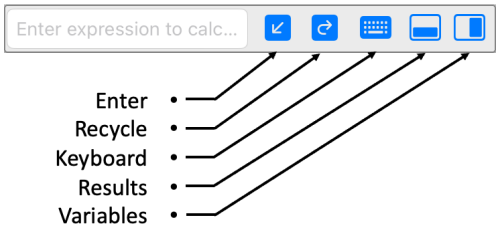


Figure 10 - Calculator Menu Bar

- The *Enter* button calculates the entered expression.
- *Recycle* will re-enter the previous expressions.

    ⇒ NOTE: Only saves the last 100 expression entries during app's session. This is reset whenever the app is restarted.

- *Keyboard* will show or hide the keyboard panel while entering an expression.
- *Results* will show or hide the *Results* area display output.
- *Variables* will show or hide the *Variables* column area.

1.2.2.1 Extended Expression Editing Mode

If the Preference's *Extended Expression Entry* setting is enabled then the following panel will be only displayed in the *Calculator* tab on iPad devices. This is intended to work with the Apple's Pencil device and the keyboard panel entering an expression to calculate.

Figure 11 - Extended Calculator's Expression Entry

- moves the insertion cursor to the left.
- moves the insertion cursor to the right.
- moves the insertion cursor to the beginning of the text.
- moves the insertion cursor to the end of the text.
- select all the text.
- extend text selection to the left.
- extend text selection to the right.
- () insert () characters.
- clear expression text field.
- erase everything in the *Calculator* tab.
- {} pop-up menu collection of units to be inserted into the text field.
- pop-up menu collection of trigonometry functions to be inserted into the text field.
- √x pop-up menu collection of math and miscellaneous functions to be inserted into the text field.
- $ pop-up menu collection of financial functions to be inserted into the text field.
- # pop-up menu collection of mathematical constants to be inserted into the text field.

⇒ NOTE: On the iPhone device, it will display a subset of the buttons in the keyboard bar.

Using the Pencil requires reasonably good penmanship. It takes a little practice to use it effectively. However, you will encounter mis-translation while entering an expression. Here a collection of hints using the Pencil:

- Draw a vertical line between characters in the text to join or split the text.
- Draw line through the text to select it.
- Draw line up-n-down over the text to delete it. If the scratching is too spaced out as drawing a line though it will prompt a *Cut*, *Copy*, and etc dialog.
- Draw `2` may translate to `2`, `z`, or `Z` character. Likewise, `x` may translate `x` or `X` character.
- Draw *sin(x)* with the dot in the *i* character will translate to lowercase *sin* characters. Whereas, drawing *cos(x)* will translate to uppercase *COS* characters.
- Drawing mathematical expression will not translate, such as, x^2, $\frac{1}{x}$, $\sqrt{x}$, and etc. Basically, it will only translate characters found in a standard alpha-numeric keyboard.
- Draw × multiple and ÷ divide character will not translate, instead use `*` and `/` characters respectively.

To enable Pencil on the iPad device, run the *Settings* app, select *Apple Pencil* and enable *Scribble* switch. Also, select *Try Scribble* button to learn how to use the pencil.

1.2.2.2 Commands

Command	Description
Ctrl+R	Evaluate expression
Ctrl+C	Clear output results
Ctrl+Option+↓	Select next evaluated expression
Ctrl+Option+↑	Select previous evaluated expression
Ctrl+Option+←	Select top evaluated expression
Ctrl+Option+→	Select bottom evaluated expression
Ctrl+1	Switch to Calculator tab
Ctrl+2	Switch to Outputs tab
Ctrl+3	Switch to Help tab
−	The delete button allows removing an entry
⩧	The sort button allows sorting the column
⁝≡	The order button allows reordering the entries

Table 1 - Calculator Commands

1.2.3 Outputs

The *Outputs* tab has an output area that displays results of the expressions entered on the *Calculator* tab and various informational, warning, and error messages.

⇒ NOTE: The output is limited and based on the *Getting Started* sub-section in the *Menu Bar* section *Settings* on the *Outputs (KB)* slider setting.

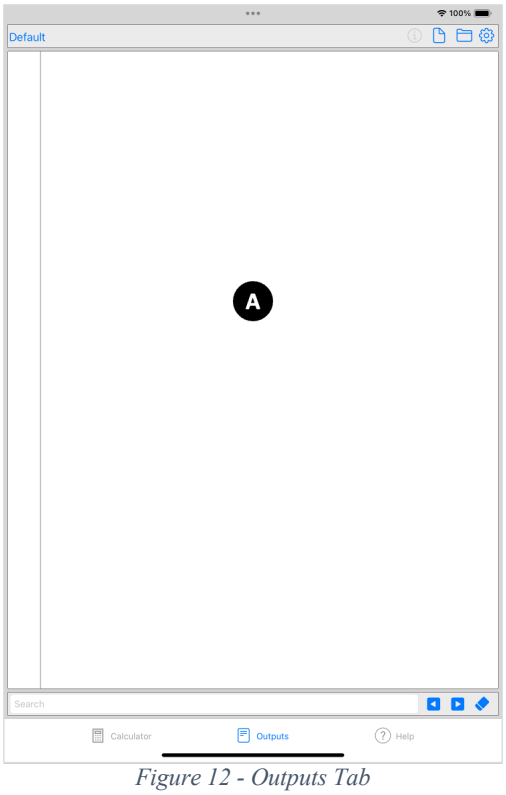


Figure 12 - Outputs Tab

The bottom part of the *Outputs* tab displays basic search-and-clear output results controls.

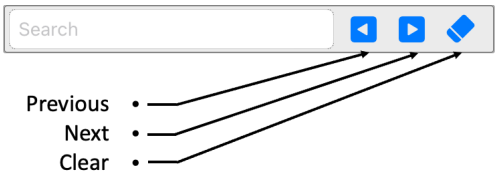


Figure 13 - Outputs Search-and-Clear Bar

- *Previous* searches for previous matching text.

    ⇒ NOTE: If there is no text in the *Search* text field, it will scroll to the previous page of the output results. If the button is double-tapped, it will scroll to the top of the results output.

- *Next* searches for next matching text.

    ⇒ NOTE: If there is no text in the *Search* text field then it will scroll to the next page of the output results. If the button is double-tapped, it will scroll to the bottom of the results output.

- *Clear* clears all output logging messages.

    ⇒ NOTE: If the button is double-tapped, the *Calculator* tab's *Results* area will also be cleared.

1.2.3.1 Commands

Command	Description
Ctrl+S	Search for next matching text
Ctrl+Shift+S	Search for previous matching text
Ctrl+C	Clear the outputted results
Ctrl+Shift+↓	Scroll to next page
Ctrl+Shift+↑	Scroll to previous page
Ctrl+Shift+←	Scroll to the top of the outputted results
Ctrl+Shift+→	Scroll to the bottom of the outputted results
Ctrl+1	Switch to Calculator tab
Ctrl+2	Switch to Outputs tab
Ctrl+3	Switch to Help tab

Table 2 - Outputs Commands

1.2.4 Help

The *Help* tab displays the app's built-in document for viewing purposes.

⇒ NOTE: You do not need a cellular or Wi-Fi connection to read the app's included documentation.

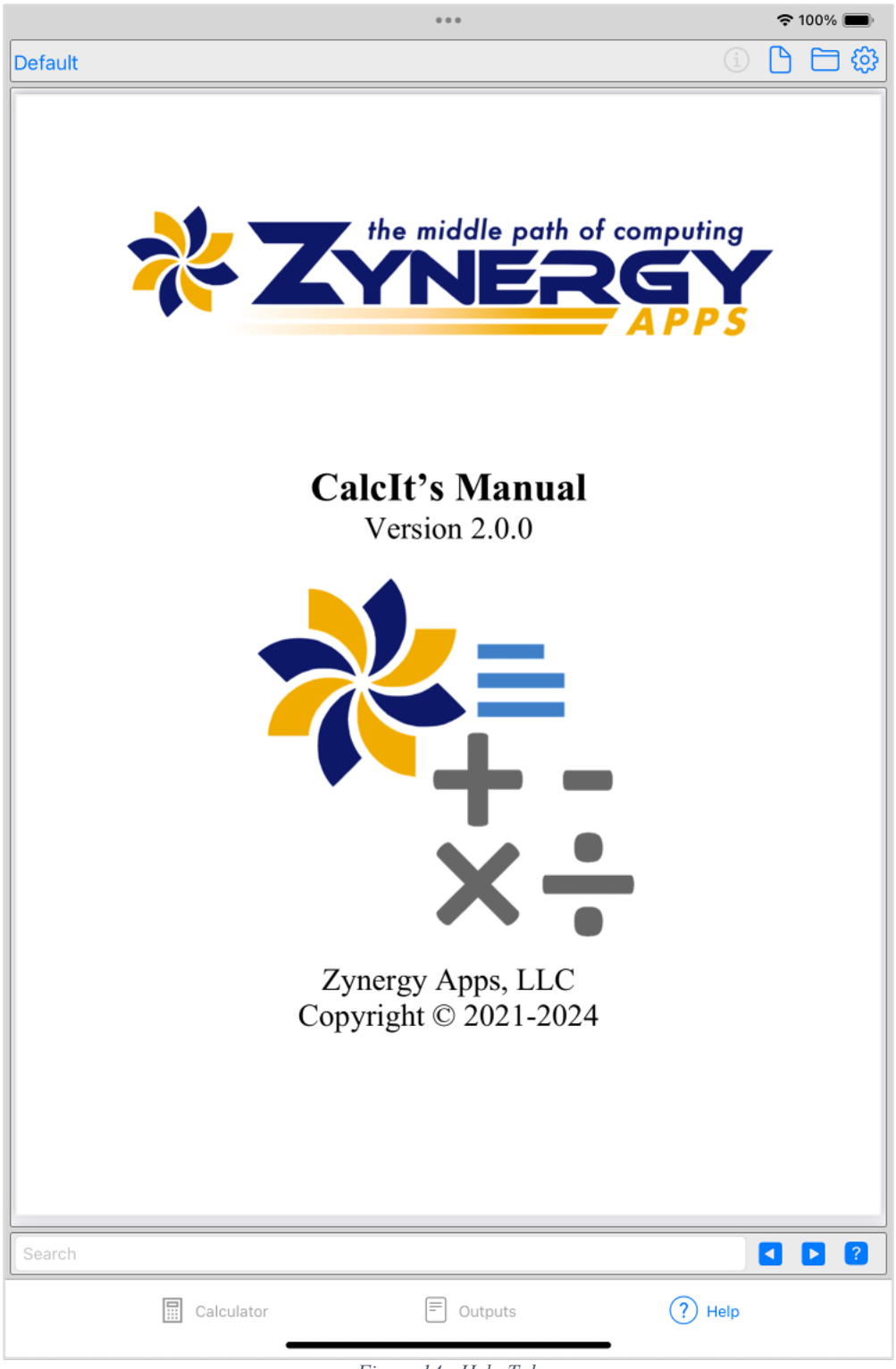


Figure 14 - Help Tab

The bottom part of the *Help* tab displays basic search controls.

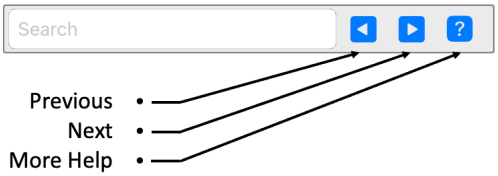


Figure 15 - Help Search Bar

- *Previous* searches for previous matching text.

    ⇒ NOTE: If there is no text in the *Search* text field, it will scroll to the previous page of the documentation. If the button is double-tapped, it will scroll to the top of the documentation.

- *Next* searches for next matching text.

    ⇒ NOTE: If there is no text in the *Search* text field, it will scroll to the next page of the documentation. If the button is double-tapped, it will scroll to the bottom of the documentation.

- *More Help* displays a dialog panel of additional documentations of the app and various mathematical subjects.

1.2.4.1 Commands

Command	Description
Ctrl+S	Search for next matching text
Ctrl+Shift+S	Search for previous matching text
Ctrl+Shift+↓	Scroll to next page
Ctrl+Shift+↑	Scroll to previous page
Ctrl+Shift+←	Scroll to the top of the documentation
Ctrl+Shift+→	Scroll to the bottom of the documentation
Ctrl+1	Switch to Calculator tab
Ctrl+2	Switch to Outputs tab
Ctrl+3	Switch to Help tab

Table 3 - Help Commands

1.2.5 Projects

The *Projects* panel has two sections. Section **A** is the *Summary* column, displaying the project's expressions and variables. Section **B** is the *Names* column, displaying a list of projects.

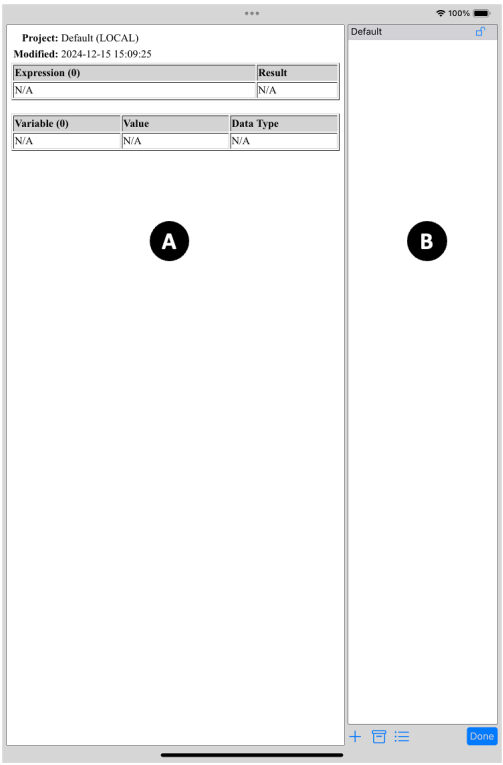


Figure 16 - Projects Dialog

The sub-menu bar at the bottom of the panel contains several button commands.

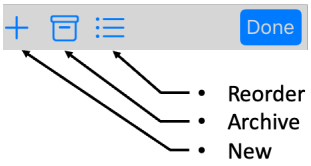


Figure 17 - Projects Menu Bar

- *Reorder* allows reordering the project's *Names* column.
- *Archive* displays *Archive* panel to restore previous saved projects or delete archived projects.
- *New* prompts you for new project name to be added.
- *Done* closes this panel and returns to the main screen of the app.

1.3 Demo Tour

This demonstration will tour the major components of the app to give you an overall understanding of its features and capabilities.

To create a *Demo* project, perform the following steps:

1.		Click the *Project* button in the top-level menu.
2.		Click the *Manage…* pop-up menu item to display the *Project* panel.
3.	+	Click *New* at the bottom of the panel's sub-menu bar to create a new project.

4.		The *New Project* dialog will be displayed.
5.		Enter *Demo* in the *Name* text field.
6.		Save the project locally on Apple mobile devices by keeping the *Save project on iCloud* option deselected.
7.		Click *OK* to create the project; you will see *Demo* added to the *Names* column.
8.		Click *Done* at the bottom of the panel's sub-menu bar to return the main screen.

Table 4 - Create Demo Project

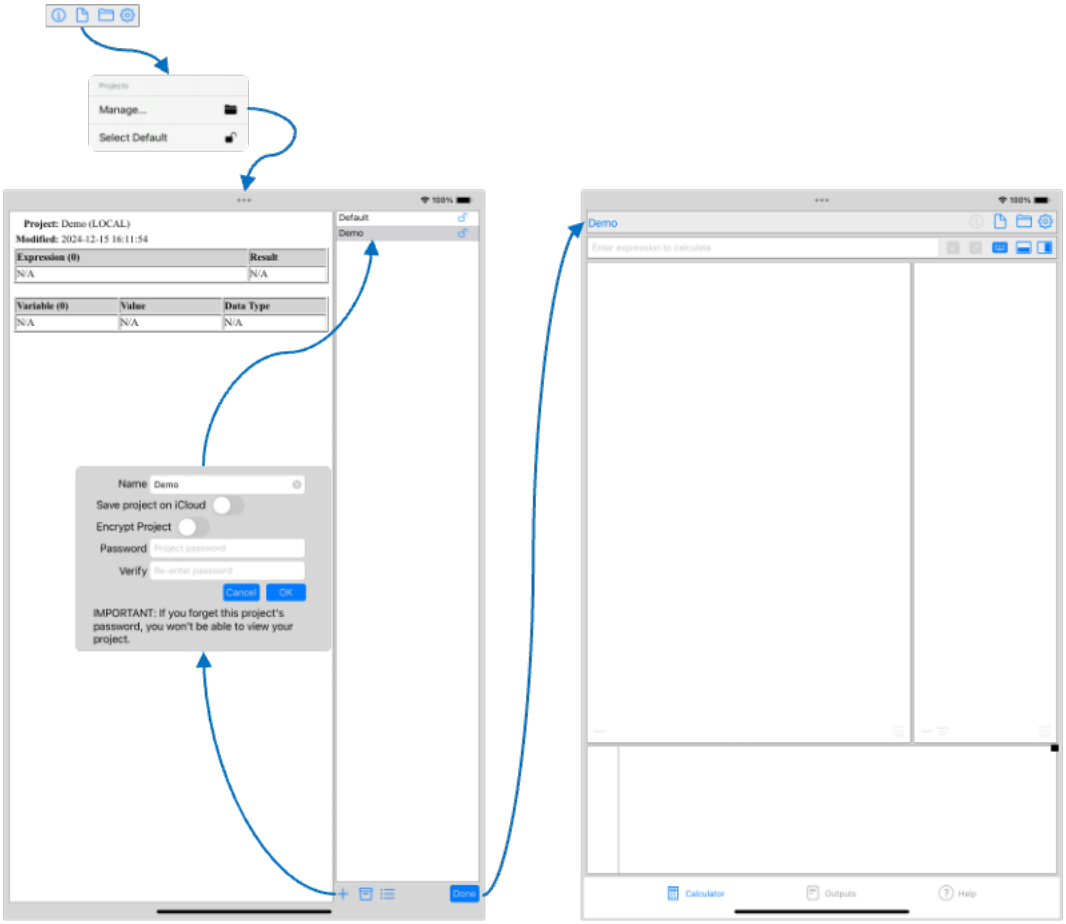


Figure 18 - Create Demo Project

⇒ NOTE: To create an encrypted project select *Encrypt Project* option and enter the password. Please if you forget this project's password, you won't be able to view your project.

To install expression examples, perform the following steps:

1.	⚙	Click the *Settings* button in the top-level menu bar to display the *Configuration* dialog.
2.		Click *Install Examples* pop-up menu item to display the *Examples* dialog.
3.		Click *Yes* to populate the *Demo* project's collection of expression examples.

Table 5 - Install Advance Script Examples

Figure 19 - Install Advance Script Examples

To manage projects, perform the following steps:

1.		Click *Project* in the top-level menu bar.
2.		Click the *Manage…* pop-up menu item to display the *Project* panel.
3.		Click *Demo* project in the *Names* column and slide it to the left.
4.		*Mail* will e-mail the project file. This will invoke the iOS Mail app to e-mail the project file.
5.		Click *Rename* button to rename the *Demo* and the *Rename "Demo" File* dialog will be displayed. For this exercise, click *Cancel*. Conversely, in the immediate dialog, you can also duplicate the dataset, too.
6.		Click *Archive* to archive the *Demo* project and the dialog will be displayed.
7.		Enter `Demonstration Project` in the *Title*'s text field.
8.		Enter `Save Demo project for future use` in the *Comment*'s text field.
9.		Click *OK* to archive the *Demo* project.
10.		Click *Default* project in the *Names* column.
11.		Slide *Demo* entry to the left without selecting it. ⇒ NOTE: You can't delete a project if it's currently used.
12.		Click *Delete* to delete the *Demo* project and it will display *Delete Project* dialog.
13.		Click *Yes* to delete the *Demo* project.

Table 6 - Renaming, Archiving, and Deleting Projects

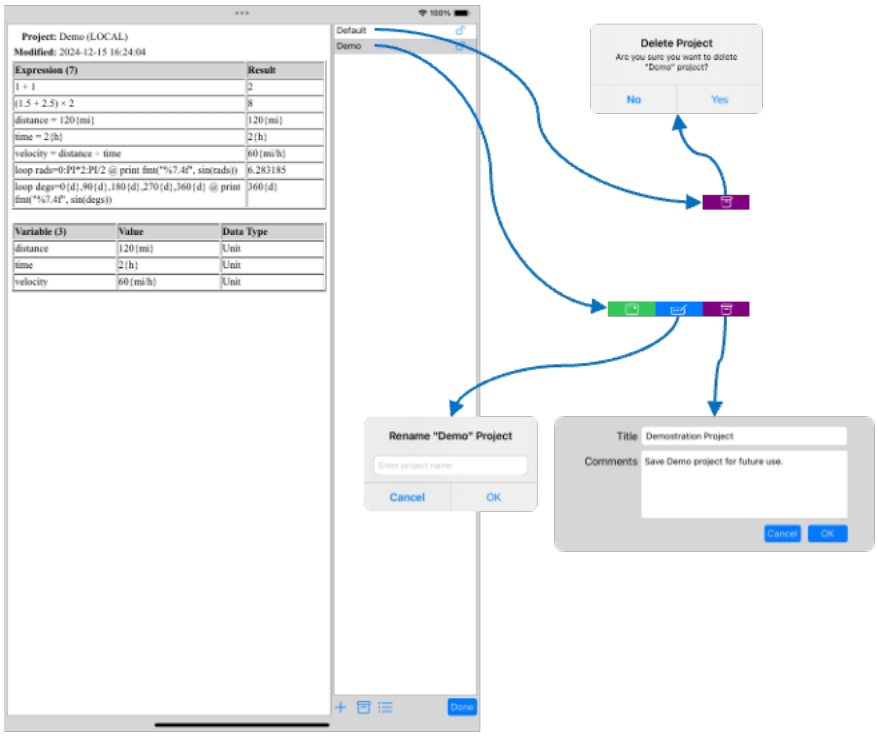


Figure 20 - Renaming, Archiving, and Deleting Projects

To restore or delete archives, perform the following steps:

1.		Click *Archive* in the sub-menu bar at the bottom right of the *Projects* panel to display the *Archive* panel.
2.		Select the second entry *Type* containing *Trash* and slide it to the left.
3.		Click *Trash* to delete the archive and the *Delete Archive* dialog will be displayed. For this exercise, select *No*.
4.		Select the first entry *Type* containing *Archive* and slide it to the left.
5.		Click *Archive* to restore the *Demo* project and the *Restore Archive* dialog will be displayed.
6.		Click *Local* to restore the *Demo* project on your device.
7.		Click *Done* to return *Projects* panel.
8.		Click *Done* again to return the main screen.

Table 7 - Restoring and Removing Archived Projects

⇒ NOTE: There are three archival types: `ARCHIVE`, `TRASH`, and `BACKUP`:

- The `ARCHIVE` type is user-initiated action to back up the project.
- The `TRASH` type is a copy of the project prior deleting it from the app. Trash backups are deleted after 7 days.
- The `BACKUP` type is a copy of the project prior to new changes to be saved into the project. The last 10 copies of the backup are retained. The oldest remainder are deleted from the app.

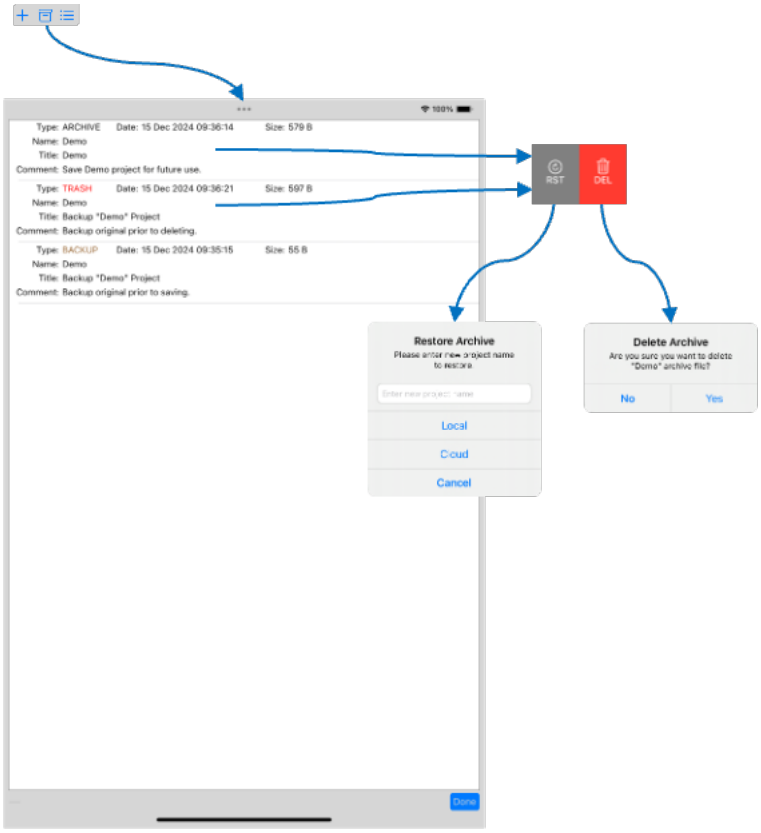


Figure 21 - Restoring and Removing Archived Projects

2 Expressions

2.1 Variables

A variable name consists of letters (i.e., a-z and A-Z), digits (i.e., 0-9) and the underscore character. The first character must be a letter. It is case-sensitive, so variable `x` is different from variable `X`.

⇒ NOTE: There is no limit to how long the variable name can be. It is generally recommended to make the name descriptive of the purpose of the variable.

There are two types of variables: dynamic and declared. The dynamic variables are not specifically declared as `integer`, `real`, `unit`, or `string` types. It depends on the assignment to the variable that inherits the data type. Dynamic variable is the default type.

For example:

- `x = 1` Variable `x` contains an integer value of `1`.
- `x = 1.5` Contains a real number value of `1.5`.

Declared variables implicitly define the type of variable it will be.

For example:

- `Integer x = 1` — Variable `x` will always contain an integer value containing initialized value `1`.
- `x = 1.5` — Real value of `1.5` will be truncated into an integer value of `1`.

2.2 Data Types

There are four fundamental data types: `integer`, `real`, `unit`, and `string`. The `unit` type is not typical data types found in other computing scripts or languages, such as C/C++, Java, Swift, and others; these are special types and will be covered in sub-sections below.

2.2.1 Integer

An integer number can be represented in several ways. The simplest of integer values are `0`, `141`, `314159`, `-27315`, `+662607015`, etc. There are other representations of integers to help with the readability of large numbers, as shown in the table below.

The integer can contain a period character on every third digit from right to left. For example, `1.000.000` is equivalent to `1,000,000` in traditional representation of 1 million. Refer to the table below for more examples.

If the integer is followed by `k`, `m`, `b`, `t`, or `q`, it's a multiplier of the value for shorthand notations. This can be lowercase or uppercase.

- `k` is thousands
- `m` is millions
- `b` is billions
- `t` is trillions
- `q` is quadrillions

Integer	Value	Comments
`1.000.000`	`1000000`	million
`1.000.000.000`	`1000000000`	billion
`1.000.000.000.000`	`1000000000000`	trillion
`1.000.000.000.000.000`	`1000000000000000`	quadrillion
`1k`	`1000`	thousand
`1.000k`	`1000000`	million
`1m`	`1000000`	million
`1.000m`	`1000000000`	billion
`1b`	`1000000000`	billion
`1.000b`	`1000000000000`	trillion
`1t`	`1000000000000`	trillion
`1.000t`	`1000000000000000`	quadrillion
`1q`	`1000000000000000`	quadrillion

Table 8 - Alternate Integer Representations

⇒ NOTE:
- If you use `1.00`, `1.000`, `1.00000`, etc., this is real number value of `1.0`.

- `1.000` is an exception of using a period with three subsequent digits defaulting to real number `1.0`, rather than `1,000`. This is because this is ambiguous between an integer or real number.

⇒ QUIRK: Dividing integer numbers can yield some unexpected results.

- If you enter the following:

```
Integer a = 1000
Integer b = 1000000
Integer c = a ÷ b
```

  The `c` variable will result to zero, because you are explicitly using integers.

- If you enter `1000 ÷ 1000000`, the result will be promoted to a real data type of `0.001`. Refer to the *String*

String is a simple data type container of sequence of characters. You can perform simple concatenation of strings into a single string value.

For example:

- `"Hello"` A simple string containing the word Hello
- `'World'` Alternate way of defining a string using single-quote character
- `"Hello" + " " + "World"` Concatenates strings to create a new `"Hello World"` string

⇒ NOTE: If one is performing numerical arithmetic operation with the string, it will attempt to convert the string to a number before applying the operation; otherwise, it will convert the numerical value to string, then concatenate.

- Mixed-Type Promotions sub-section on data type promotion rules.

2.2.2 Real

A floating-point number can be represented a couple of ways. The simplest real values are `0.1`, `1.41`, `3.14159`, `-273.15`, `+6.62607015`, etc.

For larger numbers, using powers of tens, the syntax is `[-+] <digits>.<factional> e[-+] <exponent>`.

For example:

- `6.62607015 × 10-34` is represented as `6.62607015e-34`, where `e` is the exponent character followed by numeric powers of tens.

- The exponent of `34` means moving the decimal point `34` digits to the right, which is typically zero, which is a large number. The maximum exponent is `304`.

⇒ QUIRK: If you attempt to operate a large number with a small number, it can exhibit unexpected results, but it's correct in the floating-point realm. This is due to the

device's computation limitations. This is normal in the computing world. There is wealth of science on this subject, which is beyond the scope of this manual. You just need to be aware of these obtuse conditions if your results are not what you expected. A key point to remember is that a floating-point number cannot hold an infinitely small or large value.

- If you enter `6.62607015e+34 + 1`, the result is `6.62607015e+34` because the precision is only `17` digits and power of ten makes up the rest of the floating-point value.

    - It's like saying, *`6626070150000000`*`00000000000000000 + 1 =` *`6626070150000000`*`00000000000000001`, but when the powers of tens are applied that `1` value is lost. Nothing is perfect.

2.2.3 Unit

A unit number is a real number associated with a unit description, such as length, time, and energy. Assigning and operating on these numbers will convert from one unit to another like converting feet to miles and vice-versa.

The unit number syntax is `[-+] <number> {<unit>}`, where `<number>` is the real number and `<unit>` is description of the units.

For example:

- `120{mi}` `120` miles
- `2{h}` `2` hours
- `60{mi/h}` Velocity of `60` miles per hour
- `120{mi} ÷ 2{h}` Computes `60{mi/h}`

When adding and subtracting units, they both must be the same type to allow the conversions. That is, the *Linear* category will allow converting between feet and miles.

For example:

- `5{mi} + 5280{ft}` Computes `6{mi}`
- `5280{ft} + 5{mi}` Computes `31680{ft}`

When multiplying and dividing units, the category's units add or subtract each unit numerator and denominator accordingly.

For example:

- `120{mi} ÷ (2{h} × 2{h})` Units will compute `60{mi/h^2}`, meaning acceleration of `60` miles per hour square

2.2.3.1 Categories

There are 13 categories: *Linear*, *Time*, *Angle*, *Frequency*, *Temperature*, *Power*, *Voltage*, *Amperage*, *Farad*, *Energy*, *Light*, *Weight*, and *Liquid*. The table below lists all the unit combinations available.

Category	Acronym	Description
Linear	in	inch
	ft	foot
	kft	kilofoot
	yd	yard
	mi	mile
	nmi	nautical mile
	rod	rod
	mm	millimeter
	cm	centimeter
	dm	decimeter
	m	meter
	km	kilometer
Time	ps	picosecond
	ns	nanosecond
	ms	millisecond
	s	second
	min	minute
	h	hour
	da	day
	wk	week
	mn	month
	yr	year
Angle	d	degree
	r	radian
	mr	milliradian
	gr	gradient
Frequency	hz	hertz
	khz	kilohertz
	mhz	megahertz
	ghz	gigahertz
Temperature	k	kelvin
	c	celsius
	f	fahrenheit
Power	w	watt
	uw	microwatt
	mw	milliwatt
	kw	kilowatt
Voltage	v	volt
	uv	microvolt
	mv	millivolt
	kv	kilovolt
Amperage	i	amperage
	ui	micro-amperage
	mia	milli-amperage
	ki	kilo-amperage
Farad	fa	farad

Category	Acronym	Description
	pfa	pico-farad
	ufa	micro-farad
	mfa	milli-farad
	kfa	kilo-farad
Energy	erg	energy
	j	joule
	ev	electron volt
	kwh	kilowatt hour
Weight	oz	ounce
	lb	pound
	g	gram
	mg	milligram
	kg	kilogram
Liquid	l	liter
	ga	gallon

Table 9 - Unit Categories

2.2.4 String

String is a simple data type container of sequence of characters. You can perform simple concatenation of strings into a single string value.

For example:

- `"Hello"` A simple string containing the word Hello
- `'World'` Alternate way of defining a string using single-quote character
- `"Hello" + " " + "World"` Concatenates strings to create a new `"Hello World"` string

⇒ NOTE: If one is performing numerical arithmetic operation with the string, it will attempt to convert the string to a number before applying the operation; otherwise, it will convert the numerical value to string, then concatenate.

2.2.5 Mixed-Type Promotions

The primary feature of the app is it not requiring the specifying of the data type. However, depending on how the calculations are performed, one may need to understand the various data type conversions that can affect the results.

The table below describes data type conversions. The symbol ⊗ is an operator for arithmetic operations: + (add), − (subtract), × (multiply), ÷ (divide), and % (modulo). The ⟸ is an operation result of the ⊗ operator.

Convert		Left		Right	Comments
integer, real	⟸	integer	+	integer	If addition exceeds maximum integer value, result is converted to real.
integer	⟸	integer	−	integer	
integer, real	⟸	integer	×	integer	If multiplication exceeds maximum integer value, result is converted to real.
real	⟸	integer	÷	integer	
real	⟸	integer	⊗	real	

Convert		Left		Right	Comments
unit	⟸	integer	⊗	unit	
	⟸	integer	⊗	string	Invalid operation
real	⟸	real	⊗	integer	
real	⟸	real	⊗	real	
unit	⟸	real	⊗	unit	
	⟸	real	⊗	string	Invalid operation
unit	⟸	unit	⊗	integer	
unit	⟸	unit	⊗	real	
unit	⟸	unit	⊗	unit	
	⟸	unit	⊗	string	Invalid operation
integer	⟸	string	⊗	integer	If string is not a numerical value, it's zero.
real	⟸	string	⊗	real	If string is not a numerical value, it's zero.
unit	⟸	string	⊗	unit	If string is not a numerical value, it's zero.
string	⟸	string	+	string	The + add operator is only applicable for strings.

Table 10 - Data Conversions

2.3 Arithmetic Operators

The arithmetic expression used to calculate a standard set of operators is found on typical calculators consisting of +, −, ×, ÷, %, and ^ characters. The calculations ordering depends on the precedence order, meaning higher precedence is done first, down to the lowest, as shown in the table below. Use parentheses `(` and `)` to change the order of the precedence's computation.

For example:

- `1 + 2 × 3` — The multiply operator has higher precedence by first multiplying `2 × 3`, then adding 1, resulting in `7`.
- `(1 + 2) × 3` — Using the parentheses changes of the order of precedence by first adding `1 + 2`, then multiplying by `3`, and resulting in `6`.
- `22 % 4` — The modulo operator first takes `22`, divides by `4`, uses the whole-number portion of the result, and multiplies by `4`, then subtracts it from `22`, resulting in `2`. The resulting effect of modulo `4` ranges from `0` to `3`.
- `10^3` — The power operator will perform 10^3 (i.e., `10×10×10`), resulting in `1000`.

Operator	Precedence	Description
+	1	Add
−	1	Subtract
×	2	Multiply can use alternate character: *
÷	2	Divide can use alternate character: /
%	2	Modulo
^	3	Exponentiation

Table 11 - Arithmetic Operators

2.4 Unary Operators

The unary operator will convert the value of the expression.

Operator	Precedence	Description
+	4	Doesn't change the expression's result (i.e., no operation).
−	4	Converts a positive expression's result to a negative value. And, conversely, negative expression's result to a positive value.

Table 12 - Unary Operators

3 Commands

3.1 Loop

The `loop` command iterates the `expression` statement `n`-times.

The syntax is as follows:

```
loop variable = list @ expression
```

The *`variable`* is current value looping though a *`list`* of values. The *`list`* is a series of numerical values or arithmetic expressions assigned to the *`variable`*.

The *`expression`* will be executed for each *`variable`*'s value in the `loop`.

For example:

```
loop i = 1:6 @ print "i: ", i
```

The result of the command will print out the variable's `i` value `6` times.

⇒ NOTE: The maximum number of times the expression can be executed is 250 times.

3.2 Print

The `print` command will print the list arguments on the statement.

The syntax is as follows:

```
print expression, …
```

The `print` command will print out the results of each *`expression`* in the list separated by commas. Refer to the ***Loop*** section for an example of using the `print` command.

4 Libraries

The purpose of this chapter is to briefly describe all the functions without going into detail about trigonometry, etc. There are many venues in which to learn these mathematical concepts.

4.1 Math

4.1.1 Trigonometry

⇒ LINK: https://en.wikipedia.org/wiki/Trigonometry

4.1.1.1 cos

`cos(x)` – Returns cosine angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is `0 ≤ x ≤ 2π`. One radian is equivalent to `180 ÷ PI` degrees. Alternately, use `57.2958{d}` degrees, which is equivalent to `1{r}` radian.

For example:

- `cos(0)` Returns `1`
- `cos(PI)` Returns `-1`
- `cos(180{d})` Returns `-1`

4.1.1.2 sin

`sin(x)` – Returns sine angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is `0 ≤ x ≤ 2π`.

For example:

- `sin(0)` Returns `0`
- `sin(PI)` Returns `0`
- `sin(180{d})` Returns `0`

4.1.1.3 tan

`tan(x)` – Returns tangent angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is `0 ≤ x ≤ π`.

For example:

- `tan(0)` Returns `0`
- `tan(PI ÷ 4)` Returns `1`
- `tan(90{d})` Returns `1`

4.1.1.4 acos

`acos(x)` – Returns inverse cosine's principal value back to radians

The `x` argument is the principal value. The range of `x` is $-1 \leq x \leq 1$.

For example:

- `acos(-1)` Returns `3.141593` (i.e., π)
- `acos(1)` Returns `0`

4.1.1.5 asin

`asin(x)` – Returns inverse sine's principal value back to radians

The `x` argument is the principal value. The range of `x` is $-1 \leq x \leq 1$.

For example:

- `asin(-1)` Returns `-1.570796` (i.e., $\pi \div 2$)
- `asin(1)` Returns `1.570796` (i.e., $\pi \div 2$)

4.1.1.6 atan

`atan(x)` – Returns inverse tangent's principal value back to radians

The `x` argument is the principal value. The range of `x` is $-\infty < x < \infty$.

For example:

- `atan(-1000000)` Returns `-1.570795` (i.e., ~ $\pi \div 2$)
- `atan(1000000)` Returns `1.570795` (i.e., ~ $\pi \div 2$)

4.1.1.7 cosh

`cosh(x)` – Returns hyperbolic cosine's value

The range of `x` is $-\infty < x < \infty$.

For example:

- `cosh(-2)` Returns `3.762196`
- `cosh(2)` Returns `3.762196`

4.1.1.8 sinh

`sinh(x)` – Returns hyperbolic sine's value

The range of `x` is `-∞ < x < ∞`.

For example:

- `sinh(-2)` Returns `-3.62686`
- `sinh(2)` Returns `3.62686`

4.1.1.9 tanh

`tanh(x)` – Returns hyperbolic tangent's value

The range of `x` is `-∞ < x < ∞`.

For example:

- `tanh(-1.5)` Returns `-0.9051482536448664`
- `tanh(1.5)` Returns `0.9051482536448664`

4.1.1.10 acosh

`acosh(x)` – Returns hyperbolic arc-cosine's value

The range of `x` is `1 < x < ∞`.

For example:

- `acosh(1)` Returns `0`
- `acosh(3)` Returns `1.762747`

4.1.1.11 asinh

`asinh(x)` – Returns hyperbolic arc-sine's value

The range of `x` is `-∞ < x < ∞`.

For example:

- `asinh(1)` Returns `-1.818446`
- `asinh(3)` Returns `1.818446`

4.1.1.12 atanh

`atanh(x)` – Returns hyperbolic arc-tangent's value

The range of `x` is `-1 < x < 1`.

For example:

- `atanh(-0.5)` Returns `-0.54930614433405489`
- `atanh(0.5)` Returns `0.54930614433405489`

4.1.1.13 sec

`sec(x)` – Returns secant angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is $-\pi \div 2 < x < \pi \div 2$.

For example:

- `sec(0)` Returns `1`
- `sec(PI)` Returns `-1`

4.1.1.14 csc

`csc(x)` – Returns cosecant angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is $0 < x < \pi$.

For example:

- `csc(PI ÷ 2)` Returns `1`
- `csc(3 × PI ÷ 2)` Returns `-1`

4.1.1.15 cot

`cot(x)` – Returns cotangent angle of `x` radians

The `x` argument is the angle expressed in radians. The periodic range of `x` is $0 < x < \pi$.

For example:

- `cot(PI ÷ 2)` Returns `0`
- `cot(3 × PI ÷ 2)` Returns `0`

4.1.1.16 sech

`sech(x)` – Returns hyperbolic secant's value

The range of `x` is `-∞ < x < ∞`.

For example:

- `sech(-5)` Returns `0.01347528222130456`
- `sech(5)` Returns `0.01347528222130456`

4.1.1.17 csch

`csch(x)` – Returns hyperbolic cosecant's value

The range of `x` is `-∞ < x < 0` and `0 < x < ∞`.

For example:

- `csch(-5)` Returns `-0.01347650583058909`
- `csch(5)` Returns `0.01347650583058909`

4.1.1.18 coth

`coth(x)` – Returns hyperbolic cotangent value

The range of `x` is `-∞ < x < 0` and `0 < x < ∞`.

For example:

- `coth(-5)` Returns `-1.000091`
- `coth(5)` Returns `1.000091`

4.1.2 Exponential

4.1.2.1 exp

`exp(x)` – Returns natural exponential value

The range of `x` is `-∞ < x < ∞`.

⇒ LINK: https://en.wikipedia.org/wiki/Exponential_function

For example:

- `exp(-1)` Returns `0.36787944117144233`
- `exp(0)` Returns `1`

4.1.2.2 log

`log(x)` – Returns logarithm base `2` value

The range of `x` is `0 < x < ∞`.

⇒ LINK: https://en.wikipedia.org/wiki/Logarithm

For example:

- `log(1)` Returns `0`
- `log(2)` Returns `0.69314718055994529`

4.1.2.3 log10

`log10(x)` – Returns logarithm base `10` value

The range of `x` is `0 < x < ∞`.

For example:

- `log10(1)` Returns `0`
- `log10(2)` Returns `0.3010299956639812`

4.1.3 Miscellaneous

4.1.3.1 abs

`abs(x)` – Returns absolute value

The `x` argument is converted from negative value to positive value.

For example:

- `abs(-1)` Returns `1`
- `abs(-1.5)` Returns `1.5`
- `abs(-1.5{ft})` Returns `1.5{ft}`
- `abs("-1.5")` Returns 1.5

4.1.3.2 pow

`pow(x, p)` – Returns multiplicity of value to its power

The `x` argument is multiplied to the number of `p` argument.

For example:

- `pow(5, 2)` Returns `25`

- `pow(5.5, 2.5)` Returns `70.942538`

4.1.3.3 sqrt

`sqrt(x)` – Returns square root value

The square root of the argument's `x` is calculated.

For example:

- `sqrt(2)` Returns `1.414214`

4.1.4 Range

4.1.4.1 ceil

`ceil(x)` – Returns rounded-up integral value of x

For example:

- `ceil(2.1)` Returns `3`
- `ceil(2.5)` Returns `3`
- `ceil(2.9)` Returns `3`
- `ceil(-2.1)` Returns `-2`
- `ceil(-2.5)` Returns `-2`
- `ceil(-2.9)` Returns `-2`

4.1.4.2 floor

`floor(x)` – Returns rounded-down integral value of x

For example:

- `floor(2.1)` Returns `2`
- `floor(2.5)` Returns `2`
- `floor(2.9)` Returns `2`
- `floor(-2.1)` Returns `-3`
- `floor(-2.5)` Returns `-3`
- `floor(-2.9)` Returns `-3`

4.1.4.3 round

`round(x)` – Returns nearest rounded integral value of x

For example:

- `round(2.1)` Returns `2`

- `round(2.5)` Returns `3`
- `round(2.9)` Returns `3`
- `round(-2.1)` Returns `-2`
- `round(-2.5)` Returns `-3`
- `round(-2.9)` Returns `-3`
- `round([-2.1,-2.5,-2.9])` Returns `-2`, `-3`, and `-3`

4.1.4.4 min

`min(`x_1`, `x_2`, ... `x_n`)` – Returns maximum value of `x`'s

The minimum value of x_1, x_2, ... x_n is returned.

For example:

- `min(1, 2)` Returns `1`
- `min(1, 2, 3)` Returns `1`

4.1.4.5 max

`max(`x_1`, `x_2`, ... `x_n`)` – Returns maximum value of `x`'s

The maximum value of x_1, x_2, ... x_n is returned.

For example:

- `max(1, 2)` Returns `3`
- `max(1, 2, 3)` Returns `3`

4.1.5 Constants

4.1.5.1 PI

`PI` – π constant

The floating-point π constant is `3.14159265358979323846`.

4.1.5.2 RADS

`RADS` – $\pi/180^o$ constant

The floating-point $\pi/180^o$ constant is `0.0174532925199432957`.

4.2 Convert

4.2.1 toInteger

`toInteger(x)` – Returns value of `x` as integer

The `x` argument is converted and returns an integer(s) from `real`, `unit,` or `string` data type. The fractional part of the floating-point value is truncated.

⇒ NOTE: If `x` argument is `integer` data type, it will return `x`. No conversion is performed.

For example:

- `toInteger(1.5)` Returns `1`
- `toInteger(1.5{ft})` Returns `1`
- `toInteger("1.5")` Returns `1`

4.2.2 toReal

`toReal(x)` – Returns value of `x` as floating-point

The `x` argument is converted and returns a floating-point(s) from `integer`, `unit`, or `string` data type.

⇒ NOTE: If `x` argument is `real` data type, it will return `x`. No conversion is performed.

For example:

- `toReal(1)` Returns `1.0`
- `toReal(1{ft})` Returns `1.0`
- `toReal("1")` Returns `1.0`

4.2.3 fmt

`fmt(f, x1, x2, ... xn)` – Returns formatted string of `x`'s values

The `f` argument string format is used to format the value of x_1, x_2, ... x_n is returned as a formatted string. The table below will parse `f` argument's string formatting conversion tags. Any other character sequences are pass-though.

Conversion Tags	Description
`%d`	Convert x_i to integer data type
`%<n>d`	Convert x_i to integer data type with *n* number of spaces or digits: • If *n* is preceded with `0,` pre-pad with zeros up to the *n* number of zeros. • If minus sign character precedes the *n* attribute, it will format the outputted integer to be left-justified.

	• If comma character precedes the *n* attribute, it will format the outputted integer to contain comma character on every thousandth digit.
%g	Convert x_i to floating-point data type into either integer value or integral floating-point value, floating-point value with factional part, or floating-point with exponent value
%f	Convert x_i to floating-point data type
%e	Convert x_i to floating-point data type with exponent
%<*n.m*>g, f, e	Convert x_i to floating-point data type with *n* number of digits with *m* factional digits: • If comma character precedes the *n* attribute, it will format the outputted floating-point to contain comma character on every thousandth integral digit.
%u	Convert x_i to floating-point data type with associated unit specification
%<*n.m*>u	Convert x_i to floating-point data type with *n* number of digits with *m* factional digits with associated unit specification: • If comma character precedes the *n* attribute, it will format the outputted floating-point to contain comma character on every thousandth integral digit. • If exclamation character precedes the *n* attribute, it will translate the floating-point value to integer value.
%p	Convert x_i to percentage
%<*n.m*>p	Convert x_i to percentage with *n* number of digits with *m* factional digits: • If comma character precedes the *n* attribute, it will format the outputted floating-point to contain comma character on every thousandth integral digit.
%s	Convert to string
%<*n*>s	Convert the x_i to integer data type with *n* number of spaces or digits: • If minus sign character precedes the *n* attribute, it will format the outputted string to be left-justified.
%%	Pass though % character, rather than converting the input argument

For example:

- `fmt("Hello World %d...", 2024)` Returns `Hello World 2024...`
- `fmt("'%s'", "4")` Returns `'4'`
- `fmt("'%s'", 4)` Returns `'4'`
- `fmt("'%s'", 4.5)` Returns `'4.5'`
- `fmt("'%s'", 4{d})` Returns `'4{d}'`
- `fmt("'%10s'", "4")` Returns `'         4'`
- `fmt("'%-10s'", "4")` Returns `'4         '`
- `fmt("'%10s'", 4)` Returns `'         4'`
- `fmt("'%10s'", 4.5)` Returns `'       4.5'`
- `fmt("'%10s'", 4{d})` Returns `'      4{d}'`
- `fmt("'%d'", 4)` Returns `'4'`
- `fmt("'%10d'", "4")` Returns `'         4'`
- `fmt("'%10d'", 4)` Returns `'         4'`
- `fmt("'%-10d'", 4)` Returns `'4         '`
- `fmt("'%+10d'", 4)` Returns `'        +4'`
- `fmt("'%-+10d'", 4)` Returns `'+4         '`
- `fmt("'%10d'", 4.5)` Returns `'         4'`
- `fmt("'%10d'", 4{d})` Returns `'         4'`

- `fmt("'%g'",` `4.5)` Returns `'4.5'`
- `fmt("'%10g'",` `"4")` Returns `'         4'`
- `fmt("'%10g'",` `4)` Returns `'         4'`
- `fmt("'%10g'",` `4.5)` Returns `'       4.5'`
- `fmt("'%10g'",` `4{d})` Returns `'         4'`
- `fmt("'%f'",` `4.5)` Returns `'4.500000'`
- `fmt("'%10.4f'",` `"4")` Returns `'    4.0000'`
- `fmt("'%10.4f'",` `4)` Returns `'    4.0000'`
- `fmt("'%10.4f'",` `4.5)` Returns `'    4.5000'`
- `fmt("'%10.4f'",` `4{d})` Returns `'    4.0000'`
- `fmt("'%e'",` `4.5)` Returns `'4.500000e+00'`
- `fmt("'%10.4e'",` `"4")` Returns `'4.0000e+00'`
- `fmt("'%10.4e'",` `4)` Returns `'4.0000e+00'`
- `fmt("'%10.4e'",` `4.5)` Returns `'4.5000e+00'`
- `fmt("'%10.4e'",` `4{d})` Returns `'4.0000e+00'`
- `fmt("'%u'",` `45.95)` Returns `'45.95'`
- `fmt("'%u'",` `45.95{d})` Returns `'45.95{d}'`
- `fmt("'%5.2u'",` `45.95)` Returns `'45.95'`
- `fmt("'%5.2u'",` `45.95{d})` Returns `'45.95{d}'`
- `fmt("'%8.2u'",` `45.95{d})` Returns `'45.95{d}'`
- `fmt("'%,12.2u'",` `4500.95{d})` Returns `'  4,500.9{d}'`
- `fmt("'%,d'",` `1000000)` Returns `'1,000,000'`
- `fmt("'%,7d'",` `1000000)` Returns `'1,000,000'`
- `fmt("'%,9d'",` `1000000)` Returns `'  1,000,000'`
- `fmt("'%,9.1f'",` `1000000.1)` Returns `'1,000,000.1'`
- `fmt("'%p'",` `4.5%)` Returns `'4%'`
- `fmt("'%5.2p'",` `4.5%)` Returns `' 4.50%'`
- `fmt("'%8.3p'",` `4000.5%)` Returns `'4000.500%'`
- `fmt("'%,8.0p'",` `4000.5%)` Returns `'    4,001%'`
- `fmt("'%p'",` `4000.5%)` Returns `'4001%'`

4.3 Financials

4.3.1 payment_future

`payment_future(i, n, v)` – Returns future value of periodic payments

The `i` argument is the interest rate of the periodic period, the `n` argument is the number of periods, and `v` is the periodic payments used to calculate the future value at the end of the periods.

Simple compound interest of `4%` for `15` years of `$500` monthly deposit:

- `payment_future(4%÷12, 15×12, -500)` Returns `$12,3045.24`

4.3.2 payment_periodic

`payment_periodic(i, n, v)` – Returns loan's periodic payments

The `i` argument is the interest rate of the periodic period, the `n` argument is the number of periods in the life of the loan, and `v` is the loan used to calculate the monthly payments.

Mortgage loan of `$250,000` at `4%` annual interest rate for `15` years example:

- `payment_periodic(4%÷12, 15×12, 250k)` Returns `$1,849.22`

4.3.3 payment_interest

`payment_interest(i, j, n, v)` – Returns interest portion of a periodic payment

The `i` argument is the interest rate of the periodic period, the `j` argument is the periodic period of the loan, the `n` argument is the number of periods in the life of the loan, and `v` is the loan used to calculate the interest payment portion of the period applied to the loan.

Mortgage loan interest of the first month of `$250,000` at `4%` annual interest rate for `15` years example:

- `payment_interest(4%÷12,1,15×12,250k)` Returns `$833.33`

4.3.4 payment_principal

`payment_principal(i, j, n, v)` – Returns principal portion of a periodic payment

The `i` argument is the interest rate of the periodic period, the `j` argument is the periodic period of the loan, the `n` argument is the number of periods in the life of the loan, and `v` is the loan used to calculate the principal payment portion of the period applied to the loan.

Mortgage loan principle of the first month of `$250,000` at `4%` annual interest rate for `15` years example:

- `payment_principal(4%÷12,1,15×12,250k)` Returns `$1,015.89`

⇒ NOTE: The examples above show the interest plus the principal is equal to the loan's first monthly payment.

4.4 Unit

The following table consists of all the unit conversion functions used to convert x argument's floating-point value to the equivalent unit's converted value. The input and output floating-point values are unitless. Meaning, if passed `1{ft}` to the `unit_in2ft` function it will pass the floating-point value of `1` that will be converted to `1/12` of a foot.

For example:

- `unit_in2ft(12)` Returns `1`

Category	Function	Description
Linear	`unit_in2ft(x)`	inch to feet
	`unit_in2kft(x)`	inch to kilo-feet
	`unit_in2yd(x)`	inch to yard
	`unit_in2mi(x)`	inch to mile
	`unit_in2rod(x)`	inch to rod
	`unit_in2mm(x)`	inch to millimeter
	`unit_in2cm(x)`	inch to centimeter
	`unit_in2dm(x)`	inch to decimeter
	`unit_in2m(x)`	inch to meter
	`unit_in2km(x)`	inch to kilometer
	`unit_in2nmi(x)`	inch to nautical mile
	`unit_ft2in(x)`	feet to inch
	`unit_ft2kft(x)`	feet to kilo-feet
	`unit_ft2yd(x)`	feet to yard
	`unit_ft2mi(x)`	feet to mile
	`unit_ft2rod(x)`	feet to rod
	`unit_ft2mm(x)`	feet to millimeter
	`unit_ft2cm(x)`	feet to centimeter
	`unit_ft2dm(x)`	feet to decimeter
	`unit_ft2m(x)`	feet to meter
	`unit_ft2km(x)`	feet to kilometer
	`unit_ft2nmi(x)`	feet to nautical mile
	`unit_kft2in(x)`	kilo-feet to inch
	`unit_kft2ft(x)`	kilo-feet to feet
	`unit_kft2yd(x)`	kilo-feet to yard
	`unit_kft2mi(x)`	kilo-feet to mile
	`unit_kft2rod(x)`	kilo-feet to rod
	`unit_kft2mm(x)`	kilo-feet to millimeter
	`unit_kft2cm(x)`	kilo-feet to centimeter
	`unit_kft2dm(x)`	kilo-feet to decimeter
	`unit_kft2m(x)`	kilo-feet to meter
	`unit_kft2km(x)`	kilo-feet to kilometer
	`unit_kft2nmi(x)`	kilo-feet to nautical mile
	`unit_yd2in(x)`	yard to inch
	`unit_yd2ft(x)`	yard to feet
	`unit_yd2kft(x)`	yard to kilo-feet
	`unit_yd2mi(x)`	yard to mile
	`unit_yd2rod(x)`	yard to rod
	`unit_yd2mm(x)`	yard to millimeter

Category	Function	Description
	`unit_yd2cm(x)`	yard to centimeter
	`unit_yd2dm(x)`	yard to decimeter
	`unit_yd2m(x)`	yard to meter
	`unit_yd2km(x)`	yard to kilometer
	`unit_yd2nmi(x)`	yard to nautical mile
	`unit_mi2in(x)`	mile to inch
	`unit_mi2ft(x)`	mile to feet
	`unit_mi2kft(x)`	mile to kilo-feet
	`unit_mi2yd(x)`	mile to yard
	`unit_mi2rod(x)`	mile to rod
	`unit_mi2mm(x)`	mile to millimeter
	`unit_mi2cm(x)`	mile to centimeter
	`unit_mi2dm(x)`	mile to decimeter
	`unit_mi2m(x)`	mile to meter
	`unit_mi2km(x)`	mile to kilometer
	`unit_mi2nmi(x)`	mile to nautical mile
	`unit_rod2in(x)`	rod to inch
	`unit_rod2ft(x)`	rod to feet
	`unit_rod2kft(x)`	rod to kilo-feet
	`unit_rod2yd(x)`	rod to yard
	`unit_rod2mi(x)`	rod to mile
	`unit_rod2mm(x)`	rod to milli-meter
	`unit_rod2cm(x)`	rod to centi-meter
	`unit_rod2dm(x)`	rod to decimeter
	`unit_rod2m(x)`	rod to meter
	`unit_rod2km(x)`	rod to kilometer
	`unit_rod2nmi(x)`	rod to nautical mile
	`unit_mm2in(x)`	milli-meter to inch
	`unit_mm2ft(x)`	milli-meter to feet
	`unit_mm2kft(x)`	milli-meter to kilo-feet
	`unit_mm2yd(x)`	milli-meter to yard
	`unit_mm2mi(x)`	milli-meter to mile
	`unit_mm2rod(x)`	milli-meter to rod
	`unit_mm2cm(x)`	milli-meter to centimeter
	`unit_mm2dm(x)`	milli-meter to decimeter
	`unit_mm2m(x)`	milli-meter to meter
	`unit_mm2km(x)`	milli-meter to kilometer
	`unit_mm2nmi(x)`	milli-meter to nautical mile
	`unit_cm2in(x)`	centi-meter to inch
	`unit_cm2ft(x)`	centi-meter to feet
	`unit_cm2kft(x)`	centi-meter to kilo-feet

Category	Function	Description
	`unit_cm2yd(x)`	centimeter to yard
	`unit_cm2mi(x)`	centimeter to mile
	`unit_cm2rod(x)`	centimeter to rod
	`unit_cm2mm(x)`	centimeter to millimeter
	`unit_cm2dm(x)`	centimeter to decimeter
	`unit_cm2m(x)`	centimeter to meter
	`unit_cm2km(x)`	centimeter to kilometer
	`unit_cm2nmi(x)`	centimeter to nautical mile
	`unit_dm2in(x)`	decimeter to inch
	`unit_dm2ft(x)`	decimeter to feet
	`unit_dm2kft(x)`	decimeter to kilo-feet
	`unit_dm2yd(x)`	decimeter to yard
	`unit_dm2mi(x)`	decimeter to mile
	`unit_dm2rod(x)`	decimeter to rod
	`unit_dm2mm(x)`	decimeter to millimeter
	`unit_dm2cm(x)`	decimeter to centimeter
	`unit_dm2m(x)`	decimeter to meter
	`unit_dm2km(x)`	decimeter to kilometer
	`unit_dm2nmi(x)`	decimeter to nautical mile
	`unit_m2in(x)`	meter to inch
	`unit_m2ft(x)`	meter to feet
	`unit_m2kft(x)`	meter to kilo-feet
	`unit_m2yd(x)`	meter to yard
	`unit_m2mi(x)`	meter to mile
	`unit_m2rod(x)`	meter to rod
	`unit_m2mm(x)`	meter to millimeter
	`unit_m2cm(x)`	meter to centimeter
	`unit_m2dm(x)`	meter to decimeter
	`unit_m2km(x)`	meter to kilometer
	`unit_m2nmi(x)`	meter to nautical mile
	`unit_km2in(x)`	kilometer to inch
	`unit_km2ft(x)`	kilometer to feet
	`unit_km2kft(x)`	kilometer to kilo-feet
	`unit_km2yd(x)`	kilometer to yard
	`unit_km2mi(x)`	kilometer to mile
	`unit_km2rod(x)`	kilometer to rod
	`unit_km2mm(x)`	kilometer to millimeter
	`unit_km2cm(x)`	kilometer to centimeter
	`unit_km2dm(x)`	kilometer to decimeter
	`unit_km2m(x)`	kilometer to meter
	`unit_km2nmi(x)`	kilometer to nautical mile

Category	Function	Description
	`unit_nmi2in(x)`	nautical mile to inch
	`unit_nmi2ft(x)`	nautical mile to feet
	`unit_nmi2kft(x)`	nautical mile to kilo-feet
	`unit_nmi2yd(x)`	nautical mile to yard
	`unit_nmi2mi(x)`	nautical mile to mile
	`unit_nmi2rod(x)`	nautical mile to rod
	`unit_nmi2mm(x)`	nautical mile to millimeter
	`unit_nmi2cm(x)`	nautical mile to centimeter
	`unit_nmi2dm(x)`	nautical mile to decimeter
	`unit_nmi2m(x)`	nautical mile to meter
	`unit_nmi2km(x)`	nautical mile to kilometer
Time	`unit_ps2ns(x)`	pico-second to nanosecond
	`unit_ps2ms(x)`	pico-second to millisecond
	`unit_ps2s(x)`	pico-second to second
	`unit_ps2min(x)`	pico-second to minute
	`unit_ps2h(x)`	pico-second to hour
	`unit_ps2da(x)`	pico-second to day
	`unit_ps2wk(x)`	pico-second to week
	`unit_ps2mn(x)`	pico-second to month
	`unit_ps2yr(x)`	pico-second to year
	`unit_ns2ps(x)`	nano-second to picosecond
	`unit_ns2ms(x)`	nano-second to millisecond
	`unit_ns2s(x)`	nano-second to second
	`unit_ns2min(x)`	nano-second to minute
	`unit_ns2h(x)`	nano-second to hour
	`unit_ns2da(x)`	nano-second to day
	`unit_ns2wk(x)`	nano-second to week
	`unit_ns2mn(x)`	nano-second to month
	`unit_ns2yr(x)`	nano-second to year
	`unit_ms2ps(x)`	millisecond to picosecond
	`unit_ms2ns(x)`	millisecond to nanosecond
	`unit_ms2s(x)`	millisecond to second
	`unit_ms2min(x)`	millisecond to minute
	`unit_ms2h(x)`	millisecond to hour
	`unit_ms2da(x)`	millisecond to day
	`unit_ms2wk(x)`	millisecond to week
	`unit_ms2mn(x)`	millisecond to month
	`unit_ms2yr(x)`	millisecond to year
	`unit_s2ps(x)`	second to picosecond
	`unit_s2ns(x)`	second to nanosecond
	`unit_s2ms(x)`	second to millisecond

Category	Function	Description
	`unit_s2min(x)`	second to minute
	`unit_s2h(x)`	second to hour
	`unit_s2da(x)`	second to day
	`unit_s2wk(x)`	second to week
	`unit_s2mn(x)`	second to month
	`unit_s2yr(x)`	second to year
	`unit_min2ps(x)`	minute to picosecond
	`unit_min2ns(x)`	minute to nanosecond
	`unit_min2ms(x)`	minute to millisecond
	`unit_min2s(x)`	minute to second
	`unit_min2h(x)`	minute to hour
	`unit_min2da(x)`	minute to day
	`unit_min2wk(x)`	minute to week
	`unit_min2mn(x)`	minute to month
	`unit_min2yr(x)`	minute to year
	`unit_h2ps(x)`	hour to picosecond
	`unit_h2ns(x)`	hour to nanosecond
	`unit_h2ms(x)`	hour to millisecond
	`unit_h2s(x)`	hour to second
	`unit_h2min(x)`	hour to minute
	`unit_h2da(x)`	hour to day
	`unit_h2wk(x)`	hour to week
	`unit_h2mn(x)`	hour to month
	`unit_h2yr(x)`	hour to year
	`unit_da2ps(x)`	day to pico-second
	`unit_da2ns(x)`	day to nanosecond
	`unit_da2ms(x)`	day to millisecond
	`unit_da2s(x)`	day to second
	`unit_da2min(x)`	day to minute
	`unit_da2h(x)`	day to hour
	`unit_da2wk(x)`	day to week
	`unit_da2mn(x)`	day to month
	`unit_da2yr(x)`	day to year
	`unit_wk2ps(x)`	week to pico-second
	`unit_wk2ns(x)`	week to nanosecond
	`unit_wk2ms(x)`	week to millisecond
	`unit_wk2s(x)`	week to second
	`unit_wk2min(x)`	week to minute
	`unit_wk2h(x)`	week to hour
	`unit_wk2da(x)`	week to day
	`unit_wk2mn(x)`	week to month

Category	Function	Description
	`unit_wk2yr(x)`	week to year
	`unit_mn2ps(x)`	month to pico-second
	`unit_mn2ns(x)`	month to nanosecond
	`unit_mn2ms(x)`	month to millisecond
	`unit_mn2s(x)`	month to second
	`unit_mn2min(x)`	month to minute
	`unit_mn2h(x)`	month to hour
	`unit_mn2da(x)`	month to day
	`unit_mn2wk(x)`	month to week
	`unit_mn2yr(x)`	month to year
	`unit_yr2ps(x)`	year to pico-second
	`unit_yr2ns(x)`	year to nanosecond
	`unit_yr2ms(x)`	year to millisecond
	`unit_yr2s(x)`	year to second
	`unit_yr2min(x)`	year to minute
	`unit_yr2h(x)`	year to hour
	`unit_yr2da(x)`	year to day
	`unit_yr2wk(x)`	year to week
	`unit_yr2mn(x)`	year to month
Frequency	`unit_ghz2mhz(x)`	gigahertz to megahertz
	`unit_ghz2khz(x)`	gigahertz to kilohertz
	`unit_ghz2hz(x)`	gigahertz to hertz
	`unit_mhz2ghz(x)`	megahertz to gigahertz
	`unit_mhz2khz(x)`	megahertz to kilohertz
	`unit_mhz2hz(x)`	megahertz to hertz
	`unit_khz2ghz(x)`	kilohertz to gigahertz
	`unit_khz2mhz(x)`	kilohertz to megahertz
	`unit_khz2hz(x)`	kilohertz to hertz
	`unit_hz2ghz(x)`	hertz to gigahertz
	`unit_hz2mhz(x)`	hertz to megahertz
	`unit_hz2khz(x)`	hertz to kilohertz
Power	`unit_w2mw(x)`	watt to milliwatt
	`unit_w2kw(x)`	watt to kilowatt
	`unit_w2uw(x)`	watt to microwatt
	`unit_mw2w(x)`	milliwatt to watt
	`unit_mw2kw(x)`	milliwatt to kilowatt
	`unit_mw2uw(x)`	milliwatt to microwatt
	`unit_kw2w(x)`	kilowatt to watt
	`unit_kw2mw(x)`	kilowatt to milliwatt
	`unit_kw2uw(x)`	kilowatt to microwatt
	`unit_uw2w(x)`	microwatt to watt

Category	Function	Description
	`unit_uw2mw(x)`	microwatt to milliwatt
	`unit_uw2kw(x)`	microwatt to kilowatt
Voltage	`unit_v2mv(x)`	volt to millivolt
	`unit_v2kv(x)`	volt to kilovolt
	`unit_v2uv(x)`	volt to microvolt
	`unit_mv2v(x)`	millivolt to volt
	`unit_mv2kv(x)`	millivolt to kilovolt
	`unit_mv2uv(x)`	millivolt to microvolt
	`unit_kv2v(x)`	kilovolt to volt
	`unit_kv2mv(x)`	kilovolt to millivolt
	`unit_kv2uv(x)`	kilovolt to microvolt
	`unit_uv2v(x)`	microvolt to volt
	`unit_uv2mv(x)`	microvolt to millivolt
	`unit_uv2kv(x)`	microvolt to kilovolt
Amperage	`unit_i2mia(x)`	amperage to milli-amperage
	`unit_i2ki(x)`	amperage to kilo-amperage
	`unit_i2ui(x)`	amperage to micro-amperage
	`unit_mia2i`	milli-amperage to amperage
	`unit_mia2ki(x)`	milli-amperage to kilo-amperage
	`unit_mia2ui(x)`	milli-amperage to micro-amperage
	`unit_ki2i(x)`	kilo-amperage to amperage
	`unit_ki2mia(x)`	kilo-amperage to milli-amperage
	`unit_ki2ui(x)`	kilo-amperage to micro-amperage
	`unit_ui2i(x)`	micro-amperage to amperage
	`unit_ui2mia(x)`	micro-amperage to milli-amperage
	`unit_ui2ki(x)`	micro-amperage to kilo-amperage
Farad	`unit_fa2mfa(x)`	farad to milli-farad
	`unit_fa2kfa(x)`	farad to kilo-farad
	`unit_fa2ufa(x)`	farad to micro-farad
	`unit_fa2pfa(x)`	farad to pico-farad
	`unit_mfa2fa(x)`	milli-farad to farad
	`unit_mfa2kfa(x)`	milli-farad to kilo-farad
	`unit_mfa2ufa(x)`	milli-farad to micro-farad
	`unit_mfa2pfa(x)`	milli-farad to pico-farad
	`unit_kfa2fa(x)`	kilo-farad to farad
	`unit_kfa2mfa(x)`	kilo-farad to milli-farad
	`unit_kfa2ufa(x)`	kilo-farad to micro-farad
	`unit_kfa2pfa(x)`	kilo-farad to pico-farad
	`unit_ufa2fa(x)`	micro-farad to farad
	`unit_ufa2mfa(x)`	micro-farad to milli-farad
	`unit_ufa2kfa(x)`	micro-farad to kilo-farad

Category	Function	Description
	`unit_ufa2pfa(x)`	micro-farad to pico-farad
	`unit_pfa2fa(x)`	pico-farad to farad
	`unit_pfa2mfa(x)`	pico-farad to milli-farad
	`unit_pfa2kfa(x)`	pico-farad to kilo-farad
	`unit_pfa2ufa(x)`	pico-farad to micro-farad
Energy	`unit_erg2j(x)`	energy to joule
	`unit_erg2ev(x)`	energy to electron volt
	`unit_erg2kwh(x)`	energy to kilowatt hour
	`unit_j2erg(x)`	joule to energy
	`unit_j2ev(x)`	joule to electron volt
	`unit_j2kwh(x)`	joule to kilowatt hour
	`unit_ev2erg(x)`	electron volt to energy
	`unit_ev2j(x)`	electron volt to joule
	`unit_ev2kwh(x)`	electron volt to kilowatt hour
	`unit_kwh2erg(x)`	kilowatt hour to energy
	`unit_kwh2j(x)`	kilowatt hour to joule
	`unit_kwh2ev(x)`	kilowatt hour to electron volt
Temperature	`unit_k2c(x)`	kelvin to celsius
	`unit_k2f(x)`	kelvin to fahrenheit
	`unit_c2k(x)`	celsius to kelvin
	`unit_c2f(x)`	celsius to fahrenheit
	`unit_f2k(x)`	fahrenheit to kelvin
	`unit_f2c(x)`	fahrenheit to celsius
Angle	`unit_d2r(x)`	degree to radian
	`unit_d2mr(x)`	degree to milliradian
	`unit_d2gr(x)`	degree to gradient
	`unit_r2d(x)`	radian to degree
	`unit_r2mr(x)`	radian to milliradian
	`unit_r2gr(x)`	radian to gradient
	`unit_mr2d(x)`	milliradian to degree
	`unit_mr2r(x)`	milliradian to radian
	`unit_mr2gr(x)`	milliradian to gradient
	`unit_gr2d(x)`	gradient to degree
	`unit_gr2r(x)`	gradient to radian
	`unit_gr2mr(x)`	gradient to milliradian
Weight	`unit_oz2lb(x)`	ounce to pound
	`unit_oz2mg(x)`	ounce to milligram
	`unit_oz2g(x)`	ounce to gram
	`unit_oz2kg(x)`	ounce to kilogram
	`unit_lb2oz(x)`	pound to ounce
	`unit_lb2mg(x)`	pound to milligram

Category	Function	Description
	`unit_lb2g(x)`	pound to gram
	`unit_lb2kg(x)`	pound to kilogram
	`unit_mg2oz(x)`	milligram to ounce
	`unit_mg2lb(x)`	milligram to pound
	`unit_mg2g(x)`	milligram to gram
	`unit_mg2kg(x)`	milligram to kilogram
	`unit_g2oz(x)`	gram to ounce
	`unit_g2lb(x)`	gram to pound
	`unit_g2mg(x)`	gram to milligram
	`unit_g2kg(x)`	gram to kilogram
	`unit_kg2oz(x)`	kilogram to ounce
	`unit_kg2lb(x)`	kilogram to pound
	`unit_kg2mg(x)`	kilogram to milligram
	`unit_kg2g(x)`	kilogram to gram
Liquid	`unit_l2ga(x)`	liter to gallon
	`unit_ga2l(x)`	gallon to liter

Table 13 - Unit Conversions

5 Technical Notes

5.1 Download User Manual

Download this user's manual at https://zynergyapps.com/docs/CalcIt-2.0.0.pdf.

5.2 Programming Laws

Here is a collection of software, hardware, and related discipline laws that one often encounters over course of years software engineering in designing and developing apps, web pages, databases, testing, integrating, and documenting. Ones start with a vision and one hopes it doesn't degrade into ton of kludges that forces rewrite/redo's or worse trying making things to work without breaking code elsewhere in long forgotten in pile of hundreds of thousand lines of code. Some of these, you will get it and some, well you have not gotten there. So, enjoy, and again many thanks for purchasing this app. As I always said, "Are we having fun yet?" The answer is resounding YES!

- Laws of Computer Programming:
  - Any given program, when running, is obsolete.
  - If a program is useless, it will have to be documented.
  - If a program is useful, it will have to be changed.
  - Any program will expand to fill any available memory.
  - The value of a program is proportional to the weight of its output.
  - Program complexity grows until it exceeds the capability of the programmer to maintain it.
  - Make it possible for programmers to write in English and you will find out that programmers cannot write in English.

- Hare's Law of Large Programs:

- Inside every large problem is a small problem struggling to get out.

- Golub's Law of Computerdom
  - A carelessly planned project takes three times longer to completed than expected; A Carefully planned project will take only twice as long.
  - The effort required to correct the error increases geometrically with time.

- Troutman's Programming Laws
  - If a test installation functions perfectly, all subsequent systems will malfunction.
  - Not until a program has been in production for at least six months will the most harmful error then be discovered.
  - Job control cards that cannot be arranged in improper order will be.
  - Interchangeable tapes won't.
  - If the input editor has been designed to reject all bad input, an ingenious idiot will discover a method to get bad data past it.

- Bradley's Bromide
  - If computers get too powerful, we can organize them into a committee – that will do them in.

- First Law of Scientific Progress:
  - The advance of science can be measured by the rate at which exceptions to previously held laws accumulate.
  - Corollaries:
    - Exceptions always outnumber rules.
    - There are always exceptions to established exceptions.
    - By the time one masters the exceptions, no one recalls the rules which they apply.

- Sevareid's Law:
  - The chief cause of problems is solutions.

- Seay's Law:
  - Nothing ever comes out as planned.

- Allen's Law:
  - Almost anything is easier to get into than to get out of.

- Biondi's Law:
  - If your project doesn't work, look for the part you didn't think was important.

- Eng's Principle:
  - The easier it is to do, the harder it is to change.

- Perrussel's Law:
  - There is no job so simple that it cannot be done wrong.

- Sweeney's Law:
  - The length of a progress report is inversely proportional to the amount of progress.

- Wright's First Law of Quality:

    - Quality is inversely proportional to the time left for completion of the project.
    - The length of a progress report is inversely proportional to the amount of progress.

- The Bureaucracy Principle:
    - Only a bureaucracy can fight a bureaucracy.

- Horowitz's Law:
    - Wisdom consists of knowing when to avoid perfection.

- Ely's Key to Success:
    - Create a need, and fill it.

- Baruch's Observation:
    - If all you have is a hammer, everything looks like a nail.

6 Contact

- If you have a general questions about the app, please send e-mail to mailto:support@zynergyapps.com.

- If you encountered a bug in the app, please relay details to mailto:bug@zynergyapps.com:

    - If the bug is in evaluating an expression in the *Calculator* tab, please provide a simple example encountering the artifact.

    - If the bug is in the User Interface (UI), please provide sequence of steps using the app to replicate the artifact where it is misbehaving.

- If you have any suggestions, requests, or recommendations for the app, send email to mailto:suggestions@zynergyapps.com.

7 Licenses

7.1 Agreement

LICENSED APPLICATION END USER LICENSE AGREEMENT

Apps made available through the App Store are licensed, not sold, to you. Your license to each App is subject to your prior acceptance of either this Licensed Application End User License Agreement ("Standard EULA"), or a custom end user license agreement between you and the Application Provider ("Custom EULA"), if one is provided. Your license to any Apple App under this Standard EULA or Custom EULA is granted by Apple, and your license to any Third Party App under this Standard EULA or Custom EULA is granted by the Application Provider of that Third Party App. Any App that is subject to this Standard EULA is referred to herein as the "Licensed Application." The Application Provider or Apple as applicable ("Licensor") reserves all rights in and to the Licensed Application not expressly granted to you under this Standard EULA.

a. Scope of License: Licensor grants to you a nontransferable license to use the Licensed Application on any Apple-branded products that you own or control and as permitted by the Usage Rules. The terms of this Standard EULA will govern any content, materials, or services accessible from or purchased within the

Licensed Application as well as upgrades provided by Licensor that replace or supplement the original Licensed Application, unless such upgrade is accompanied by a Custom EULA. Except as provided in the Usage Rules, you may not distribute or make the Licensed Application available over a network where it could be used by multiple devices at the same time. You may not transfer, redistribute or sublicense the Licensed Application and, if you sell your Apple Device to a third party, you must remove the Licensed Application from the Apple Device before doing so. You may not copy (except as permitted by this license and the Usage Rules), reverse-engineer, disassemble, attempt to derive the source code of, modify, or create derivative works of the Licensed Application, any updates, or any part thereof (except as and only to the extent that any foregoing restriction is prohibited by applicable law or to the extent as may be permitted by the licensing terms governing use of any open-sourced components included with the Licensed Application).

b. Consent to Use of Data: You agree that Licensor may collect and use technical data and related information—including but not limited to technical information about your device, system and application software, and peripherals—that is gathered periodically to facilitate the provision of software updates, product support, and other services to you (if any) related to the Licensed Application. Licensor may use this information, as long as it is in a form that does not personally identify you, to improve its products or to provide services or technologies to you.

c. Termination. This Standard EULA is effective until terminated by you or Licensor. Your rights under this Standard EULA will terminate automatically if you fail to comply with any of its terms.

d. External Services. The Licensed Application may enable access to Licensor's and/or third-party services and websites (collectively and individually, "External Services"). You agree to use the External Services at your sole risk. Licensor is not responsible for examining or evaluating the content or accuracy of any third-party External Services, and shall not be liable for any such third-party External Services. Data displayed by any Licensed Application or External Service, including but not limited to financial, medical and location information, is for general informational purposes only and is not guaranteed by Licensor or its agents. You will not use the External Services in any manner that is inconsistent with the terms of this Standard EULA or that infringes the intellectual property rights of Licensor or any third party. You agree not to use the External Services to harass, abuse, stalk, threaten or defame any person or entity, and that Licensor is not responsible for any such use. External Services may not be available in all languages or in your Home Country, and may not be appropriate or available for use in any particular location. To the extent you choose to use such External Services, you are solely responsible for compliance with any applicable laws. Licensor reserves the right to change, suspend, remove, disable or impose access restrictions or limits on any External Services at any time without notice or liability to you.

e. NO WARRANTY: YOU EXPRESSLY ACKNOWLEDGE AND AGREE THAT USE OF THE LICENSED APPLICATION IS AT YOUR SOLE RISK. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, THE LICENSED APPLICATION AND ANY SERVICES PERFORMED OR PROVIDED BY THE LICENSED APPLICATION ARE PROVIDED "AS IS" AND "AS AVAILABLE," WITH ALL FAULTS AND WITHOUT WARRANTY OF ANY KIND, AND LICENSOR HEREBY DISCLAIMS ALL WARRANTIES AND CONDITIONS WITH RESPECT TO THE LICENSED APPLICATION AND ANY SERVICES, EITHER EXPRESS, IMPLIED, OR STATUTORY, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES AND/OR CONDITIONS OF MERCHANTABILITY, OF SATISFACTORY QUALITY, OF FITNESS FOR A PARTICULAR PURPOSE, OF ACCURACY, OF QUIET ENJOYMENT, AND OF NONINFRINGEMENT OF THIRD-PARTY RIGHTS. NO ORAL OR WRITTEN INFORMATION OR ADVICE GIVEN BY LICENSOR OR ITS AUTHORIZED REPRESENTATIVE SHALL CREATE A WARRANTY. SHOULD THE LICENSED APPLICATION OR SERVICES PROVE DEFECTIVE, YOU ASSUME THE ENTIRE COST OF ALL NECESSARY SERVICING, REPAIR, OR CORRECTION. SOME JURISDICTIONS DO NOT ALLOW THE

EXCLUSION OF IMPLIED WARRANTIES OR LIMITATIONS ON APPLICABLE STATUTORY RIGHTS OF A CONSUMER, SO THE ABOVE EXCLUSION AND LIMITATIONS MAY NOT APPLY TO YOU.

f. Limitation of Liability. TO THE EXTENT NOT PROHIBITED BY LAW, IN NO EVENT SHALL LICENSOR BE LIABLE FOR PERSONAL INJURY OR ANY INCIDENTAL, SPECIAL, INDIRECT, OR CONSEQUENTIAL DAMAGES WHATSOEVER, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF PROFITS, LOSS OF DATA, BUSINESS INTERRUPTION, OR ANY OTHER COMMERCIAL DAMAGES OR LOSSES, ARISING OUT OF OR RELATED TO YOUR USE OF OR INABILITY TO USE THE LICENSED APPLICATION, HOWEVER CAUSED, REGARDLESS OF THE THEORY OF LIABILITY (CONTRACT, TORT, OR OTHERWISE) AND EVEN IF LICENSOR HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. SOME JURISDICTIONS DO NOT ALLOW THE LIMITATION OF LIABILITY FOR PERSONAL INJURY, OR OF INCIDENTAL OR CONSEQUENTIAL DAMAGES, SO THIS LIMITATION MAY NOT APPLY TO YOU. In no event shall Licensor’s total liability to you for all damages (other than as may be required by applicable law in cases involving personal injury) exceed the amount of five dollars ($5.00). The foregoing limitations will apply even if the above stated remedy fails of its essential purpose.

g. You may not use or otherwise export or re-export the Licensed Application except as authorized by United States law and the laws of the jurisdiction in which the Licensed Application was obtained. In particular, but without limitation, the Licensed Application may not be exported or re-exported (a) into any U.S.-embargoed countries or (b) to anyone on the U.S. Treasury Department's Specially Designated Nationals List or the U.S. Department of Commerce Denied Persons List or Entity List. By using the Licensed Application, you represent and warrant that you are not located in any such country or on any such list. You also agree that you will not use these products for any purposes prohibited by United States law, including, without limitation, the development, design, manufacture, or production of nuclear, missile, or chemical or biological weapons.

h. The Licensed Application and related documentation are "Commercial Items", as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation", as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States.

i. Except to the extent expressly provided in the following paragraph, this Agreement and the relationship between you and Apple shall be governed by the laws of the State of California, excluding its conflicts of law provisions. You and Apple agree to submit to the personal and exclusive jurisdiction of the courts located within the county of Santa Clara, California, to resolve any dispute or claim arising from this Agreement. If (a) you are not a U.S. citizen; (b) you do not reside in the U.S.; (c) you are not accessing the Service from the U.S.; and (d) you are a citizen of one of the countries identified below, you hereby agree that any dispute or claim arising from this Agreement shall be governed by the applicable law set forth below, without regard to any conflict of law provisions, and you hereby irrevocably submit to the non-exclusive jurisdiction of the courts located in the state, province or country identified below whose law governs:

> If you are a citizen of any European Union country or Switzerland, Norway or Iceland, the governing law and forum shall be the laws and courts of your usual place of residence.
>
> Specifically excluded from application to this Agreement is that law known as the United Nations Convention on the International Sale of Goods.

7.2 Third-Parties

- **Text Kit Line Numbers**
  Copyright © 2013 Mark Alldritt
  MIT License: https://opensource.org/licenses/mit-license.html

- **Icon8**
  License: https://icons8.com/license