tacotui

Release 0.1

Apr 29, 2020

Contents:

CHAPTER 1

Simple terminal coloring and text-based user interface

This Python package is meant to be a simple, light-weight, cross-platform method for manipulating the terminal and generating basic text-based user interfaces. If you just need easy control of terminal coloring, or a few simple widgets that are one step above using *input()*, this module is for you.

- Cross platform. Works directly in the Windows 10 terminal, most Linux terminals, and OSX.
- Provides easy access functions for setting terminal colors and position.
- Widgets that provide a "linear" user experience. User works with one widget at a time.
- Easy and fast to program. Instead of writing objects/classes, everything is defined using function calls.
- Different color schemes for widgets.
- Pure python, no dependencies.

While packages like curses, and its wrappers such as urwid or npyscreen, are much more powerful and flexible, this has a low learning curve and works cross-platform with no additional setup. It is meant for beginning Python programmers (aka engineers who use Python but don't have time to become computer scientists) who want a quick method for generating a usable linear user interface, with no requirements for object-oriented programming.

A single function call is all that's needed to make an entry form such as this. The function returns a list of the entered parameters.


TacoTUI Order Form
First Name: Monty
Last Name: Python
Number of tacos: 3
Tortilla ◄ Flour ►
Filling: ◄ Chicken ►
Chile: ◄ Green ►
◄ Smothered ►
DONE

CHAPTER 2

Installation

Tacotui can be installed from pip:

```
pip install tacotui
```

Source code is available from Bitbucket.

This package works with Python 3.6 or higher on terminals that support ANSI escape codes. This includes Mac terminal, Windows 10 terminal, and most Linux terminals.

Tacotui consists of three modules: *screen* for screen manipulation such as cursor positioning, *color* for setting background and foreground text colors, and *widgets* with tools such as message boxes, list selection boxes, and forms.

Run the demos to see *tacotui* in action:

```
import tacotui
tacotui.demo.color()
tacotui.demo.widgets()
tacotui.demo.plot()
tacotui.demo.game()
```

2.1 Screen Module

The *tacotui.screen* module contains functions and attributes for manipulating the terminal screen and setting the cursor position. The functions set control the screen directly, where the *screen.pos* object has string attributes containing ANSI Codes that, when printed to the terminal, control the display.

2.1.1 Basic screen functions

`tacotui.screen.`**`clear`**`()`
Clear the screen and fill with the theme background color

tacotui.screen.**clearline**()
Clear the current line and fill with the current background color

tacotui.screen.**hide_cursor**(*hide=True*)
Show or hide the cursor

Attributes *screen.scrwidth* and *screen.scrheight* provide the character width and height of the screen. These are set on startup, so manually resizing the terminal while the program is running may give weird results.

2.1.2 Printing

A few functions are included that wrap the builtin print method, allowing for printing in a specific color or centering text, without printing a newline.

tacotui.screen.**sprint**(*s*)
Print without newline and immediately update the display

Parameters **s** (*string*) – The string to print

tacotui.screen.**write**(*s*, *c=None*)
Write a string in the given color

Parameters

- **s** (*string*) – The string to print
- **c** (*string*) – Color name from the *tacotui.color* module

tacotui.screen.**centertext**(*s*, *width=80*)
Center the text, padded with spaces

Parameters

- **s** (*string*) – The text to center
- **width** – Total width of final padded string

Returns **s** – Centered string padded with spaces

Return type string

Note: This currently won't work if *s* contains escape codes since Python's *format* doesn't know how to exclude those from the string length.

2.1.3 Curosr Positioning

There are two ways to set cursor position. First is the *setxy* function.

tacotui.screen.**setxy**(*x*, *y*)
Set the cursor position, (1, 1) is top left corner

x: int Horizontal cursor position

y: int Vertical cursor position

The second method uses the *screen.pos* object which has attributes that can be directly printed to the terminal. Attribues *screen.pos.x**, *screen.pos.y** and *screen.pos.xy*_** are available, where the asterisk is replaced with an integer indicating screen position. The attribute value is a string ANSI escape code that will set the terminal position. Note that

Typical usage in an f-string:

```
screen.sprint(f'{screen.pos.xy5_3}This text starts at x=5, y=3.')
```

The *screen.pos* object also has attributes *push* and *pop* that push the current screen position onto a stack.

```
screen.sprint(f'{screen.pos.x4}{screen.push}{screen.pos.x15}x=15')
screen.sprint(f'{screen.pos.pop}x=4 again')
```

2.1.4 Coloring functions

The color functions set or reset the color applied to all subsequent prints. See *Color Module* for how to specify the colors.

`tacotui.screen.`**`setcolor`**`(`*`c`*`)`
Set the current color

Parameters **c** (*string*) – Color name from the *tacotui.color* module

`tacotui.screen.`**`resetcolor`**`()`
Reset colors to terminal default

2.2 Color Module

tacotui contains two objects for setting the color of text on the screen. *tacotui.color* is used for setting specific colors, such as 'red', or 'blue', and *tacotui.theme* for setting colors based on a theme.

2.2.1 Preset colors

The *tacotui.color* object has several predefined color attributes for printing. Each attribute is a string ANSI escape code that, when printed to the terminal, sets either the background or foreground color. Python f-strings work great for changing the color mid-line.

```
from tacotui import screen, color
screen.sprint(f'{color.RED}This is red. {color.BLUE}This is blue.')
screen.sprint(f'{color.RESET}This is the default terminal color.')
```

The colors are RED, GREEN, YELLOW, BLUE, MAGENTA, CYAN, WHITE, BLACK, BRIGHTRED, BRIGHTGREEN, BRIGHTYELLOW, BRIGHTBLUE, BRIGHTMAGENTA, BRIGHTCYAN, BRIGHTWHITE, BRIGHTBLACK. Note the actual color displayed will depend on the terminal settings.

The background color is set by prepending *BG* to the color attribute:

```
screen.sprint(f'{color.BGRED}{color.YELLOW}Yellow on red{color.RESET}')
```

Colors can be bolded or underlined by adding *_BOLD* and *_UNDERLINE* to the attribute name:

```
screen.sprint(f'{color.RED_UNDERLINE}Red and underlined.{color.RESET}')
```

2.2.2 Custom RGB colors

The ANSI code for a specific RGB color can be accessed using the *color.rgb* and *color.bgrgb* functions:

```
screen.sprint(color.rgb(173, 216, 230))
screen.sprint(color.bgrgb(69, 69, 69))
screen.sprint('Light blue on dim gray')
```

2.2.3 Theme colors

The *tacotui.theme* object contains colors named by function which can be set to different themes. Theme colors are used extensively with the *Widgets Module*. Theme color names are

- NORM: Normal text
- BORDER: Borders for boxes, messages, etc.
- HLIGHT: Text that is highlighted by the cursor
- SELECT: Text that is selected or enabled, but not highlighted
- HSELECT: Text that is selected or enabled AND highlighted
- NAME: Names, titles
- MSG: Message text
- BUTTON: Button text
- ERROR: Color for error messages

Three themes are included: *blue*, *sand*, and *default*. They can be enabled using *theme.settheme(name)*. Clear the screen immediately after setting the theme to pick up the background color.

```
from tacotui import theme
theme.settheme('sand')
screen.clear()
screen.sprint(f'{theme.MSG}Message color{theme.NORM}')
```

2.3 Widgets Module

The *tacotui.widgets* module has simple user interface widgets, such as message boxes, list selection boxes, and forms. These widgets are displayed one-at-a-time, creating a linear user interface.

All widgets use the *Theme colors* for coloring. To adjust the widget colors, the theme color must be changed.

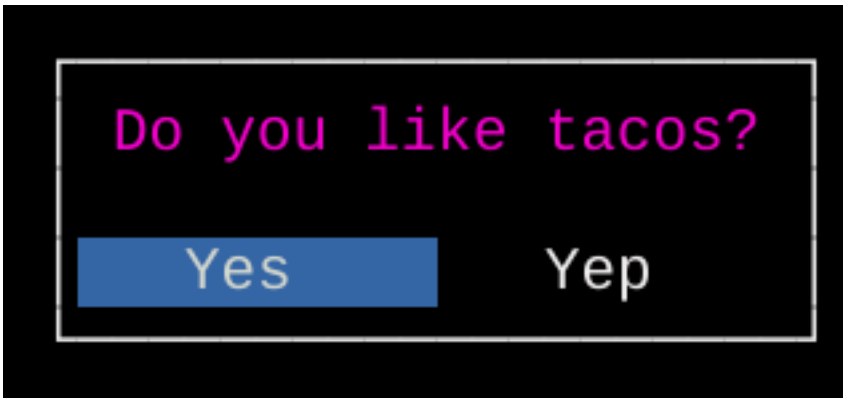

Fig. 1: Message box

tacotui.widgets.**box**(*x*, *y*, *w*, *h*)
Draw a box

Parameters

- **x** (*int*) – Left side of the box
- **y** (*int*) – Top side of the box
- **w** (*int*) – Width of the box
- **h** (*int*) – Height of the box

tacotui.widgets.**message**(*s*, *buttons=None*, ***kwargs*)
Show a message box with optional buttons

Parameters

- **s** (*string*) – Message to display
- **buttons** (*list*) – List of button names

Keyword Arguments **y** (*x,*) – Coordinates of message box

Returns **btn** – Text of the selected button, or None if no buttons are defined.

Return type string

2.3.1 Geting values from the user

The *line_edit* function returns a string entered by the user. Other widgets *get_int*, *get_float*, *get_char*, and *get_date* restrict the input to a specific type, and print an error if an invalid value is entered.

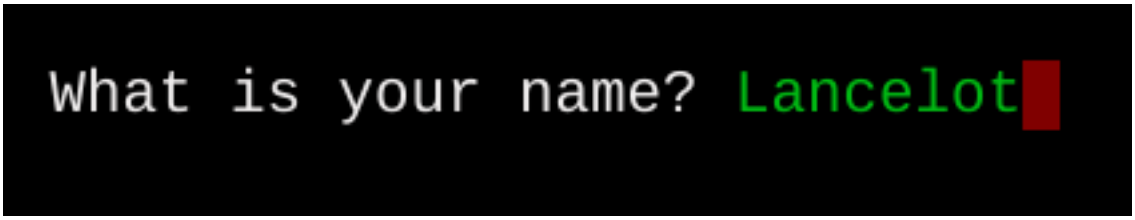

Fig. 2: Line Edit box

tacotui.widgets.**line_edit**(*s=''*, *prompt=''*, ***kwargs*)
Enter a line of text, with possible initial value

Parameters

- **s** (*string*) – Default value for line edit
- **prompt** (*string*) – Text to show in front of entry area

Keyword Arguments **y** (*x,*) – Position of line edit

Returns **value** – Line entered by the user

Return type string

tacotui.widgets.**get_int**(*prompt*, ***kwargs*)
Get an integer value.

Parameters **prompt** (*string*) – Prompt to show the user

Keyword Arguments **y** (*x,*) – Location to display prompt. An error message will display on the next line if a non-integer value is entered.

Returns **value** – Integer value entered by the user

Return type int

tacotui.widgets.**get_float**(*prompt*, ***kwargs*)
Get a float value.

Parameters **prompt** (*string*) – Prompt to show the user

Keyword Arguments

- **y** (*x,*) – Position for the prompt. If invalid or out-of-range value is entered, error message is shown on the next line.
- **fmin** (*float*) – Minimum acceptable float value
- **fmax** (*float*) – Maximum acceptable float value

Returns **value** – Floating point value entered by the user

Return type float

tacotui.widgets.**get_char**(*options=None*, ***kwargs*)
Input a character.

Keyword Arguments

- **options** (*list, optional*) – List of acceptable characters
- **y** (*x,*) – Location for showing error message if the entered character is not in options.

Returns **value** – Single-character string entered by the user

Return type string

tacotui.widgets.**get_date**(*prompt*, ***kwargs*)
Get a date-time (requires dateutil package)

Parameters

- **prompt** (*string*) – Prompt to show the user
- **y** (*x,*) – Position of prompt. If invalid date is entered, error message will show on next line.

Returns **value** – Date value entered by the user

Return type datetime

2.3.2 Selecting from a list

List select and multiselect allow selection from a predefined list of items.

tacotui.widgets.**list_select**(*options*, *prompt=''*, ***kwargs*)
Select an item from a list

Parameters

- **options** (*list*) – List of items for the list box
- **prompt** (*string*) – Text to show just above the box

Keyword Arguments

- **y** (*x,*) – Position of the box. If not provided, will be centered on the screen.

Fig. 3: List Select Widget

- **h** (*w,*) – Width and height of box. If not provided, will be determined based on the size of the options list.

Returns

- **index** (*int*) – Index of list item selected by the user
- **value** (*string*) – String of list item selected by the user

`tacotui.widgets.`**`multi_select`**(*options*, *prompt=''*, ***kwargs*)

Select multiple items from a list.

Parameters

- **`options`** (*`list`*) – List of options to show in the box
- **`prompt`** (*`string`*) – Text to show just above the box

Keyword Arguments

- **y** (*x,*) – Position of the box. If not provided, will be centered on the screen.
- **h** (*w,*) – Width and height of box. If not provided, will be determined based on the size of the options list.

Returns

- **index** (*list of int*) – List of all indexes selected by the user
- **values** (*list of string*) – List of all strings values selected by the user

2.3.3 Selecting files and folders

The *file_select* widget can be used to select an existing file, name a new file, or select an existing folder.

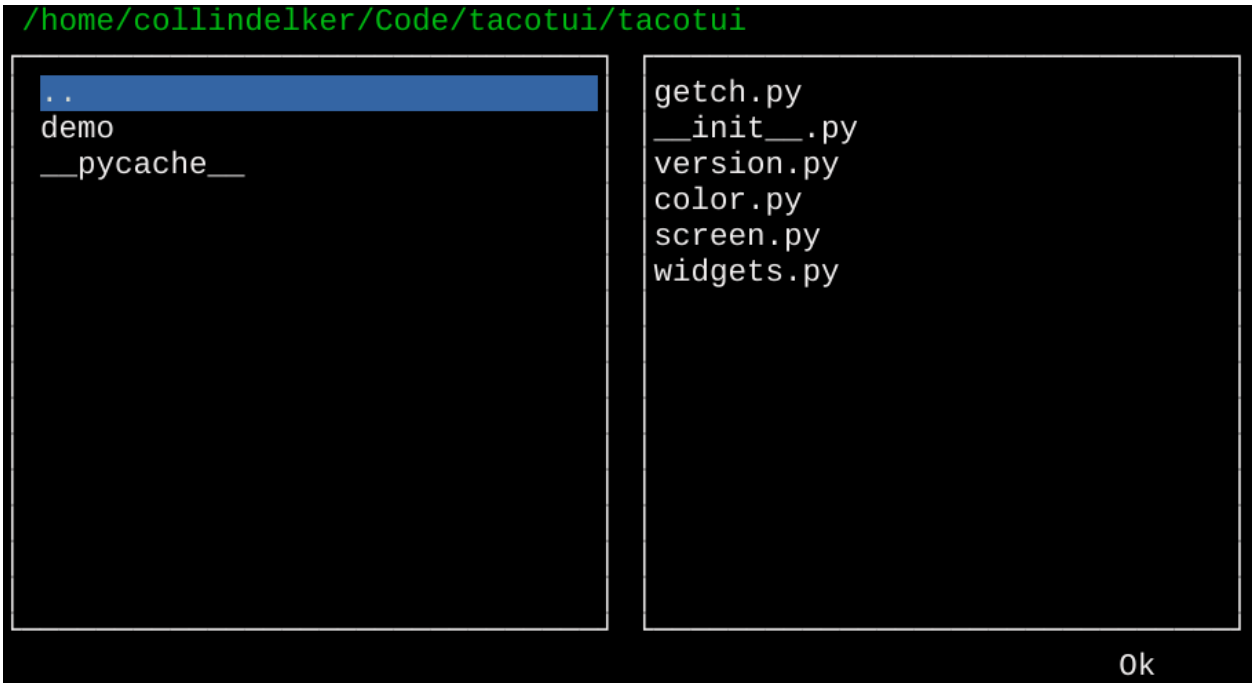

Fig. 4: File select widget

`tacotui.widgets.`**`file_select`**(*path='.'*, *mode='existing'*, ***kwargs*)

Select a file or folder.

Parameters

- **`path`** (*`string`*) – Starting folder location
- **`mode`** (*`string`*) – File selection mode. Either 'existing', 'new', or 'folder'.

Keyword Arguments

- **`y`** (*`x,`*) – Top-left position of the file selection box. Defaults to 0, 0.
- **`h`** (*`w,`*) – Width and height of file selection box. Defaults to fill the full screen.
- **`showhidden`** (*`bool`*) – Show hidden files (starting with dot)
- **`filt`** (*`iterable`*) – List of file extensions to show, example ['py', 'pyc', 'pyo']

Returns **fname** – File or folder name as entered by the user

Return type string

2.3.4 Progress and slider

`tacotui.widgets.`**`slider`**(*prompt=''*, ***kwargs*)

Make a slider widget

Parameters

- **`prompt`** (*`string`*) – Text to show above the widget

Fig. 5: Slider Widget

- **y** (*x,*) – Top-left position of the widget
- **w** (*int*) – Width in characters of the slider
- **maxval** (*minval,*) – Miniumum and Maximum value represented by the slider
- **showval** (*bool*) – Whether to print the represented value to the right of the slider
- **char** (*string*) – Character to represent slider indicator
- **color** (*string*) – Color of slider indicator
- **barcolor** (*string*) – Color of slider bar

Returns **value** – Slider value when user presses enter

Return type float

Fig. 6: Progress bar

tacotui.widgets.**progress**(*pct*, *prompt=''*, ***kwargs*)

Progress bar

Parameters

- **pct** (*float*) – Percent value to show (0-100 range)
- **prompt** (*string*) – Text to show above progress bar

Keyword Arguments

- **y** (*x,*) – Position of top left corner
- **w** (*int*) – Width of the progress bar
- **c** (*string*) – Color of indicator
- **ch** (*string*) – Character for indicator
- **showpct** (*bool*) – Print the percent value to the right of the indicator

2.3.5 Forms

The form widget allows for multiple widgets at once in a single form. The form returns a list of responses for each item.

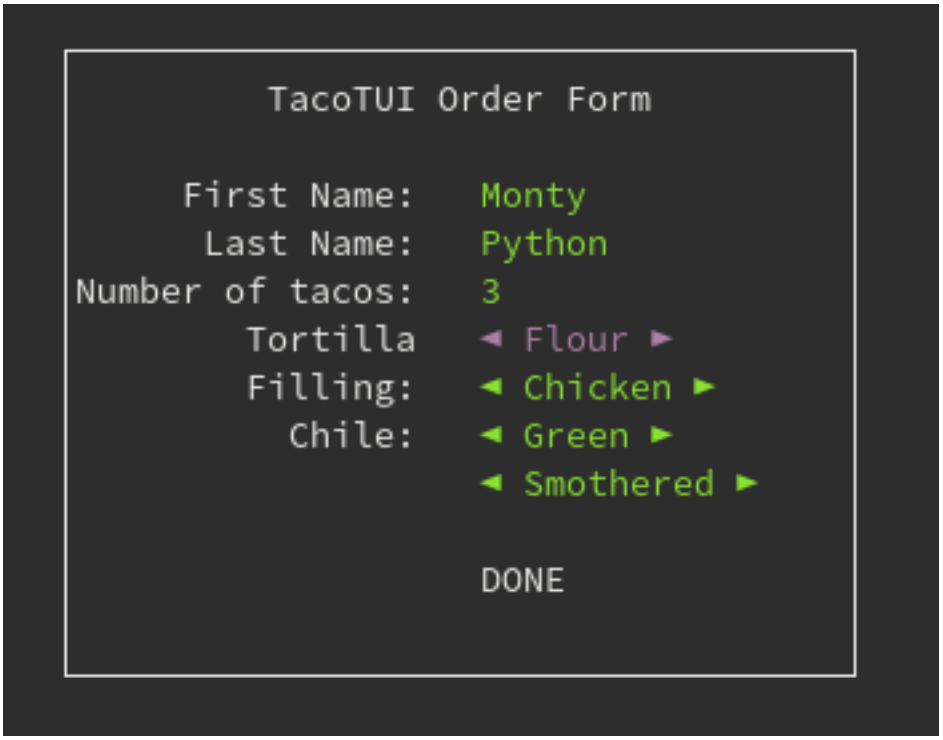

Fig. 7: Form widget

tacotui.widgets.**form**(*title=''*, **items*, ***kwargs*)

Make a form for entry of multiple items.

Parameters

- **title** (*string*) – Show a title on the form
- **items** (*tuples*) – Each row in the form is defined by a tuple of (prompt, datatype, [default], [key]) prompt: string value to show in this line datatype: either int, float, str, or a list of option values (selectable by arrow keys in the form) default (optional): initial value to show.

Keyword Arguments

- **y** (*x,*) – Top left position of the form
- **w** (*int*) – Width of the form

Returns values – List of all items entered in the form, in the same order they were defined, cast to the appropriate type.

Return type list

Notes

int and float datatypes will be validated. User cannot select DONE until they enter valid numbers (or leave them blank). floats in scientific notation are allowed.

2.3.6 Data visualization

Quick and dirty plotting is available in the command line. While limited, these functions provide basic line and bar plotting functionality without requiring large plotting packages.

tacotui.widgets.**textplot**(**xypairs*, ***kwargs*)

Plot x, y data as scatter plot

Parameters

- **xypairs** (*tuples of lists*) – Each xypair is a tuple of (x, y) data, where x and y are lists of values to plot.
- **y** (*x,*) – Top-left position of plot
- **w** (*int*) – Plot width in characters
- **h** (*int*) – Plot height in characters
- **title** (*string*) – Title to print above the plot
- **margin** (*int*) – Size of left-side margin
- **colors** (*list of string*) – List of color codes to cycle for the plot points
- **markers** (*list of string*) – List of characters to cylce for the plot points

tacotui.widgets.**barplot**(**xypairs*, ***kwargs*)

Make a horizontal bar plot

Parameters **xypairs** (*tuples*) – Tuples of (xlabel, yvalue)

Keyword Arguments

- **y** (*x,*) – Top-left position of bar plot
- **w** (*int*) – Total width of bar plot
- **char** (*string*) – Character to use for bar segment
- **showvals** (*bool*) – Print the numeric value at the end of each bar
- **colors** (*list*) – List of colors to cycle

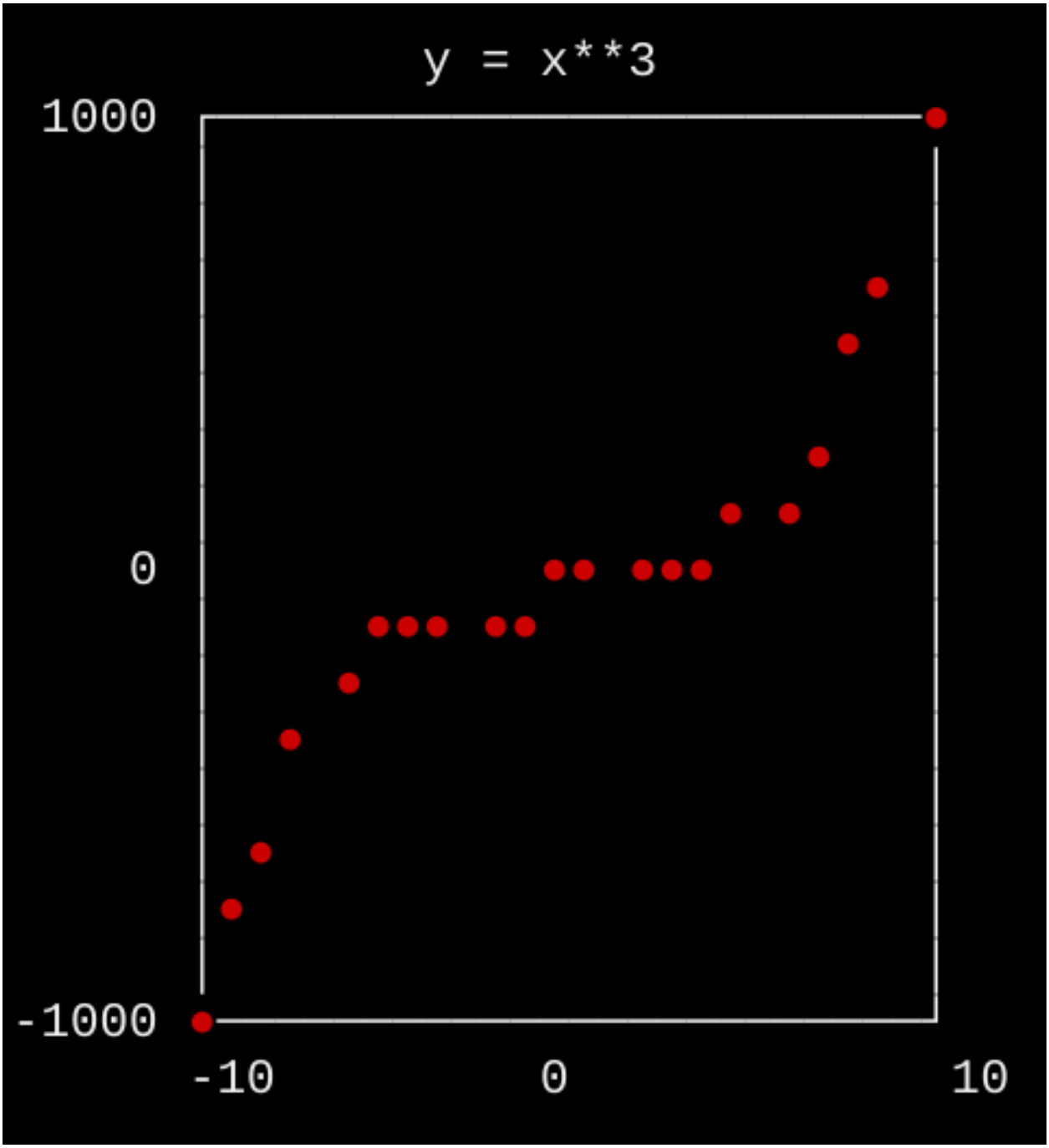

Fig. 8: Textplot

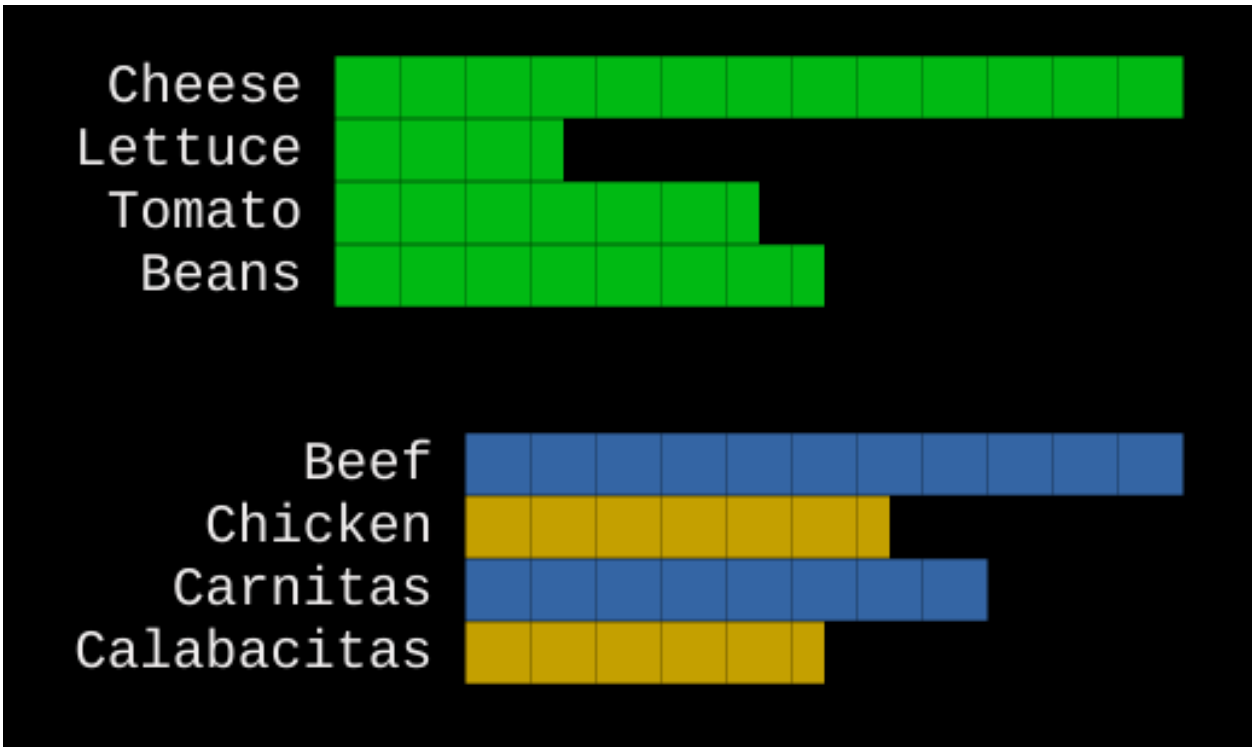

Fig. 9: Bar plot

Index