Area4 Documentation

Reece Dunham

Feb 10, 2020

CONTENTS:

CHAPTER

ONE

CONCEPT

> **Warning:** If you don't understand what we mean by dividers, fear not.

We mean dividers that divide text or sections of text in the Python console, or use cases like that.

For example:

```
    Hello World!
--------------------
Welcome to my library.
```

In this example, the line between the two sections would be the divider.

The great part about area4 is that it is built to be flexible.

We have all kinds of dividers, from simple text strings similar to the one from the example, to long emoji strings, to dividers in Markdown for programs such as GitHub bots!

Make sure to view our other documentation pages for more information!

Tip: Because area4 has a lot of emoji dividers, it can also double as an emoji database!

CHAPTER

TWO

INSTALLING

You may install in one of the following ways:

2.1 With Pip

To install via pip, open a terminal, and type one the following command:

```
# Windows:
$ pip install --upgrade area4
# macOS/Linux:
$ python3 -m pip install --upgrade area4
```

It should install. If the install fails because of a permissions error, try running the command with sudo or with the –user flag.

2.2 With a requirements.txt

To use area4 as a dependency for your project, you can add the following line:

```
area4
```

Note: You must know how to use a requirements file to take this path. If you don't, search how to use a requirements file for Python dependencies.

2.3 With Pipenv

To install with pipenv, run:

```
pipenv install area4
```

CHAPTER

THREE

USING

After you install the package (see the installing section), you need to import it into any Python file that you will use it in.

You can do this by adding the following line to the top:

```
import area4
```

Warning: Versions before 2.0 don't work with the new methods. 1.x has reached its end-of-life, and you should migrate. See the migrating guide for how to do so.

If you want to, you can check to make sure the library is working by running:

```
print(area4.area4info())
```

Now, to get dividers, use this function:

```
print(
    area4.divider(4)
)  # This prints divider number 4 to the console
```

For what all the dividers look like, see the next section.

CHAPTER

FOUR

DIVIDER LOOKS

The number before it is the number you pass to the divider function. So if you want divider 1 you would use: `area4.divider(1)`. *If you don't understand, see the examples part of the documentation.*

1. Dashed
2. Solid
3. Dotted
4. Black Squares
5. Up arrow emojis
6. Down arrow emojis
7. Equal signs
8. Hashtags
9. Asterisks (stars)
10. Commas
11. Slashes
12. Broken bars (|)
13. Tildes
14. Backslashes (not to be confused with #11)
15. Coffee cups
16. Plus signs
17. Cthulhus
18. Lenny faces
19. And (&) signs
20. Up arrow dividers (^)

21. Shrug emojis
22. Number 1s
23. Number 2s
24. Number 3s
25. Number 4s
26. Number 5s
27. Number 6s
28. Number 7s
29. Number 8s
30. Number 9s
31. Number 10s
32. <>s
33. Smiley faces I think
34. &*s
35. Random numbers (returned as string which is automatically converted from an integer)
36. The symbol that looks like ‘
37. Lowercase a
38. Uppercase a
39. Lowercase b
40. Uppercase b
41. Lowercase c
42. Uppercase c
43. Lowercase d
44. Uppercase d
45. Lowercase e
46. Uppercase e
47. Lowercase f
48. Uppercase f
49. Lowercase g
50. Uppercase g

51. Lowercase h
52. Uppercase h
53. Lowercase i
54. Uppercase i
55. Lowercase j
56. Uppercase j
57. Lowercase k
58. Uppercase k
59. Lowercase l
60. Uppercase l
61. Lowercase m
62. Uppercase m
63. Lowercase n
64. Uppercase n
65. Lowercase o
66. Uppercase o
67. Lowercase p
68. Uppercase p
69. Lowercase q
70. Uppercase q
71. Lowercase r
72. Uppercase r
73. Lowercase s
74. Uppercase s
75. Lowercase t
76. Uppercase t
77. Lowercase u
78. Uppercase u
79. Lowercase v
80. Uppercase v

81. Lowercase w
82. Uppercase w
83. Lowercase x
84. Uppercase x
85. Lowercase y
86. Uppercase y
87. Lowercase z
88. Uppercase z
89. Beach umbrella emojis
90. Airplane emojis
91. Orange leave emojis
92. Key emojis
93. Big smiles
94. Laughing face
95. Toung-sticking-out faces
96. Toung-sticking-out faces v2
97. Surprised faces
98. Upset face
99. Pretend-smile face
100. Scared face
101. Surprised face
102. Happy babies
103. Happy ladies
104. Happy men
105. Happy ladies
106. Happy men
107. Happy grannies
108. Happy grandpas
109. Thumbs up
110. Thumbs down

111. Punch
112. Fist
113. Punch left
114. Punch right
115. Crossed fingers
116. Crossed fingers v2
117. Unknown emoji
118. Ok-hand
119. Point left
120. Point right
121. Point up
122. Point down
123. Finger up
124. Hand
125. Hand v2
126. 5 fingers on hand
127. Vulcan solute emojis
128. Bye wave
129. Call me wave
130. Strong arm emoji
131. Dog emojis
132. Cat emojis
133. Mice emojis
134. Hamster emojis
135. Bunny emojis
136. Fox emojis
137. Bear emojis
138. Panda emojis
139. Koalas
140. Tigers

141. Lions
142. Cows
143. Pigs
144. Frog
145. Monkeys
146. Monkey eyes shielded
147. Monkey eyes open
148. Monkey hands over mouth
149. Sitting monkey
150. Penguins
151. Chickens
152. Parrots
153. Birds
154. Hatching ducks
155. Ducks
156. Geese
157. Flower bundles
158. Pink flowers
159. Roses
160. Dead flowers
161. Pink flowers 2
162. Pink flowers 3
163. White flower
164. Yellow flower
165. Small sun
166. Big sun
167. Half moons facing left
168. Half moons facing right
169. Full moons
170. Stars

171. Multiple stars
172. Lightning bolts
173. Water bolts
174. Fires
175. Thunder clouds
176. Rainbows
177. Partly eaten chickens
178. Not-really-eaten chickens
179. Hot dogs
180. Hamburgers
181. French fries
182. Pizza
183. Sandwiches
184. Sno-cones
185. Ice creams in cups
186. Ice creams in cones
187. Pies
188. Cakes
189. Cakes 2
190. Beers
191. Two touching beers
192. Two touching wine glasses
193. Single wine glasses
194. Soccer balls
195. Medals
196. Cars
197. Alarm clocks
198. Money bags
199. Balloons
200. Hearts

201. Pins
202. A person
203. Dice
204. Bowling ball and pins
205. Cookies
206. Snowmen
207. Potatoes
208. Shrimp
209. Hot people
210. Cold people
211. Robot emojis
212. Person having party
213. Mind blown emojis
214. Be quiet emojis
215. Semicolons
216. Eye emojis
217. Ghost emojis
218. At signs
219. Telephone emojis
220. Colons
221. Curly brackets
222. [-] emojis
223. =_=+ emojis
224. Thinking emojis
225. ⋆− dividers
226. Flower emojis
227. Persian/Arabic words stretching character
228. Percent symbols
229. Hearts (alternative to #200)
230. Negation (¬)

231. Apple logo emoji (macOS only)
232. Mountain ASCII characters
233. Upside-down mountain ASCII characters
234. Tomato emojis
235. Left brackets
236. Chili peppers
237. (Image Of symbols - see https://www.compart.com/en/unicode/U+22B7)
238. Akitas (dogs)
239. Dollar signs
240. Red dots
241. Chain links
242. Scooter emojis
243. Avacado emojis
244. 1337 dividers
245. Smiling cats
246. Smiling cats 2
247. Laughing cats
248. Heart-eyed cats
249. Mischievous cats
250. Kissing cats
251. Surprised cats
252. Sad cats
253. Mad cats
254. Top hats
255. Party poppers
256. Sponges
257. Satelite antennas
258. Links
259. Papers with pencils
260. Crossed hammers and wrenches

261. Download item icons

With more coming soon!

Thanks to amrutha3 on GitHub for making the majority of the emoji dividers, and everybody who has added a divider.

Warning: Depending on what platform the user is on, some dividers may look different. This includes some CI systems, in which emojis are not rendered in build logs.

CHAPTER

FIVE

CUSTOM DIVIDERS

You can generate a custom divider with the make_div function

```
# Specify a repeating unit and a maximum length
area4.make_div('<>', length=24)
# Returns a string

# Add start or end elements
area4.make_div('=-', length=9, start='<', end='=>')
# Returns: '<=-=-=-=>'

# Resize existing dividers
area4.make_div(area4.divider(1), length=6)
# Returns: '------'

# Setting to custom div:
custom_div = area4.make_div('<>', length=24)

# or directly printing
print(area4.make_div('<>', length=24))

# specify an literal unit (the function will not attempt to find␣
↪smaller repeating units)
area4.make_div('<><>~', length=10, literal_unit=True)
# Returns '<><>~<><>~' instead of '<><><><><>'
```

Warning: The `make_div()` function will try to replicate whole repeating units to the specified length. The output will always be less than or equal to the specified length. Test the output to ensure the divider looks as you would like it.

BIG thank you to ninexball on GitHub for making this function and maintaining it!

CHAPTER

SIX

OTHER FUNCTIONS

This is a list of other functions you may want to use, and what the do in basic terms.

See the API Refrence page for more information.

6.1 Splitter

- *area4.splitter()*

New in version 2.1.0.

The splitter function takes a string or number as a divider, and a series of strings to return, divided. If the first parameter is a number, it looks it up in the divider list. Otherwise, it uses the string provided as a divider. If only one additional string is provided, nothing is returned.

For example:

```
import area4
print(area4.splitter(1, "Welcome to", "My **app**"))

# outputs:
# Welcome to
# ------------
# My **app**
```

6.2 Get Divider Character

- area4.utils.get_divider_character()

New in version 2.1.7.

Gets you the material, or character the divider is made of. You need to pass an integer of the divider you want to get the character it is made of.

For example:

```
from area4.util import get_divider_character

print(get_divider_character(7))

# This example prints a single equal sign to the console,
# because that divider is '============'
```

6.3 Reddit Horizontal

- `area4.utils.reddit_horizontal()`

New in version 2.3.1.

This function returns the Reddit Markdown divider (for Reddit bots). This function takes no parameters.

6.4 Markdown Horizontal

- `area4.utils.markdown_horizontal()`

New in version 2.9.0.

This function returns the Markdown divider (rendered as an HTML 'hr' tag by sites like GitHub). This function takes no parameters.

Note: If you want the HTML tag equivalent of the rendered output, see *HTML Horizontal*.

6.5 HTML Horizontal

- `area4.utils.html_horizontal()`

New in version 3.1.0.

This function returns the HTML tag(s) for the divider element (rendered as a literal line by default, unless changed via CSS). This function takes 1 optional parameter, `closing_tag`, more information is detailed in the API Reference section.

CHAPTER

SEVEN

API REFERENCE

7.1 Main Module

Main module.

Copyright 2018-present Reece Dunham.

License MIT, see LICENSE for more details.

area4.**area4info**()
Get some info about the package.

Returns Package info.

Return type str

area4.**divider**(*number*)
Get the divider you requested.

Parameters **number** (*int*) – The divider number (can't be 0).

Returns The requested divider.

Return type str

Raises **ValueError** – If you request an invalid divider.

Example `area4.divider(3)` will return '.....'

area4.**make_div**(*unit*, *length=24*, *start=''*, *end=''*, *literal_unit=False*)
Generate a custom divider.

Parameters

- **unit** (*str*) – A repeating unit.
- **length** (*Optional[int]*) – The maximum length (won't be exceeded) (default: 24).
- **start** (*Optional[str]*) – Starting string.

- **end** (*Optional[str]*) – Ending string.
- **literal_unit** (*Optional[bool]*) – If `True`, it will not try to break unit down into smaller repeating subunits. Defaults to `False`.

Returns A new, custom divider.

Return type str

Example
```
custom_div = make_div(unit='=-', length=40,
    start='<', end='=>')
```

Note: The generated string will be terminated at the specified length regardless of if all the input strings have been fully replicated. A unit > 1 length may not be able to be replicated to extend to the full length. In this situation, the string will be shorter than the specified length. Example: unit of 10 characters and a specified length of 25 will contain 2 units for a total length of 20 characters.

---

area4.**splitter**(*div*, **args*)

Split text with dividers easily.

Returns The newly made value.

Return type str

Parameters **div** (*str*) – The divider.

7.2 Utilities

Utilities module.

Copyright 2018-present Reece Dunham.

License MIT, see LICENSE for more details.

area4.util.**get_divider_character**(*divider_id*)

Get the character the divider is made of.

Parameters **divider_id** (*int*) – The divider's number.

Returns The character.

Return type str

Raises **ValueError** – If you request an invalid divider.

Example Get what divider 7 is made of:

```
get_divider_character(7)
# returns '='.
```

area4.util.**get_raw_file**
Get the raw divider file in a string array.

Returns The array.

Return type List[str]

area4.util.**html_horizontal**(*closing_tag=True*)
Get HTML horizontal divider.

Parameters **closing_tag** (*Optional[bool]*) – If a closing tag should be added.

Returns The HTML tag (the divider).

Return type str

area4.util.**markdown_horizontal**()
Get Markdown horizontal divider.

Returns The divider.

Return type str

area4.util.**non_single_character_dividers**()
Get a list of all the "blacklisted" dividers.

These dividers are not made of a single character. Some examples of this include:

- Divider 18 - `( °  °)`
- Divider 33 - `^,^,^,^,^,^,`
- Divider 34 - `&*&*&*&*&*&*`

Returns A list of divider IDs.

Return type List[int]

area4.util.**reddit_horizontal**()
Get Reddit horizontal divider.

Returns The divider.

Return type str

area4.util.**reduce_to_unit**(*divider*)
Reduce a repeating divider to the smallest repeating unit possible.

This function is used by `make_div()`.

Parameters **divider** (*str*) – The divider.

Returns Smallest repeating unit possible.

Return type str

Example 'XxXxXxX' -> 'Xx'

CHAPTER

EIGHT

MIGRATING

Here are steps required to migrate to certain versions:

8.1 1.x -> 2.x

To migrate from v1, you will need to change all divider calls from:

```
area4.dividerX
# or
area4.divX()
```

Where `X` is the divider number, to:

```
area4.divider(X)
```

8.2 2.x -> 3.x

This version removed the duplicate divider (was #201), so all the dividers with numbers/IDs BIGGER than 201 need to be shifted down by 1, so if you are using any of those dividers, you will need to change the number. For example:

```
area4.divider(208)
# needs to be changed to:
area4.divider(207)

# however, anything BELOW 201 does NOT need to be changed!
```

By Reece Dunham[1]

View on GitHub

[1] <me@rdil.rocks>

Welcome to area4, the flexible divider library.

From here, you may want to proceed to the Concept page.

Want to skip right to the action? Select the Installing page.

PYTHON MODULE INDEX

a

INDEX