Area4 Documentation

RDIL

Feb 22, 2019

Contents:

CHAPTER 1

Installing

You may install in one of the following ways:

- Through pip
- Through requirements.txt

1.1 To install via pip

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

```
$ pip install area4
```

It should install.

1.2 To install via requirements.txt

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

```
area4
```

> **Warning:** You must have prior knowledge with using a requirements.txt file to take this path. If not, search how to use a requirements file for Python dependencies

CHAPTER 2

Using

After you install the package (instructions in 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
```

After doing so, you can use any of the methods listed in the examples section to get the dividers.

> **Warning:** In version 1.1.0, the functions changed in the way they operated. Before 1.1.0, the functions printed the variables, but after the release, they now just return the variable.

And if you want to you can check to make sure the library is working:

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

> **Warning:** The area4info function will only work on Python 3.6 or above. Keep this in mind if using it and you don't want SyntaxErrors!

CHAPTER 3

Divider looks

The number before it is the number used in calling it, so for example if you want divider 1, it would be area4.divider1 or area4.div1(). 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. Tildas
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 simbol 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 smilies
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.
128. Bye wave
129. Call me wave
130. Strong arm emoji
131. Dogs
132. Cats
133. Mice
134. Hampsters
135. Bunnies

136. Foxes
137. Bears
138. Pandas
139. Koalas
140. Tiger
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 v2
162. Pink flowers v3
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. Sandwitches
184. Sno-cones
185. Ice creams in cups
186. Ice creams in cones
187. Pies
188. Cakes
189. Cakes v2
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. Baloons
200. Hearts
201. Surprised faces v2
202. Pins
203. A person
204. Dice
205. Bowling ball and pins
206. Cookies
207. Snowmen
208. Potatoes
209. Shrimp

And more coming soon!

CHAPTER 4

Examples

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

We mean dividers that divide text in the Python console, or anything you use the library for.

An example can be found here.

A few examples just using plain print commands:

```
print(area4.divider1)
print(area4.divider2)
print(area4.divider3)
print(area4.divider4)
print(area4.divider5)
print(area4.divider6)
print(area4.divider7)
print(area4.divider8)
print(area4.divider9)
print(area4.divider10)
print(area4.divider11)
print(area4.divider12)
print(area4.divider13)
print(area4.divider14)
print(area4.divider15)
```

A few examples just using functions:

```
print(area4.div1())
print(area4.div2())
print(area4.div3())
print(area4.div4())
print(area4.div5())
print(area4.div6())
print(area4.div7())
print(area4.div8())
print(area4.div9())
print(area4.div10())
print(area4.div11())
print(area4.div12())
print(area4.div13())
print(area4.div14())
print(area4.div15())
```

CHAPTER 5

Custom Dividers

Important: In version 1.0.3, custom dividers were added. They can't be saved, but will stay applied until the Python script ends.

They can be called/used/updated this way:

```
# Setting:
area4.custom_div = str("divider_text_here")

# Using:
print(area4.customdiv())
# or...
print(area4.custom_div)
```

You can also 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.divider1, 6)     # Returns: '------'
```

(continues on next page)

(continued from previous page)

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

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

# specify an literal unit (the function will not atempt 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.

CHAPTER 6

Info for devs

6.1 Divider Format

The format for making new dividers is:

```
# Variable:
dividerNumber = str("dividertexthere")

# Function
def divNumber():
    return dividerNumber
```

6.2 __name__ Checking

You can check to see if __name__ is "__main__" by running this function: .. code-block:: python

> # import the utilities module: import area4.util # run function: print(area4.util.check()) # additionally, you can specify an __name__ to work with: print(area4.util.check(str(__name__)))

By RDIL[1]

View on GitHub

[1] <contactspaceboom@gmail.com>

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

Important: Please read all pages before creating issues as the question may be answered here.