

update readme.md

Browse files

master

liuzf1 committed 43 minutes ago1 parent 901cee4 commit 7217772473080aa5673b029b67f5c9929d8e0385

Showing 7 changed files with 124 additions and 118 deletions.

Split Unified

2 .gitignore

27 MANIFEST
28
29
30 - # CoderWanFeng
31 .idea/
32 .gitee/
33 shell/upload.sh

27 MANIFEST
28
29
30 + # demo
31 .idea/
32 .gitee/
33 shell/upload.sh

17 README.md

117
118 python-office欢迎任何人来添砖加瓦，贡献代
码，建议提交的pr（pull request）符合一些规
范，规范如下：
119
120 - 0. 每位参与者，请在文件夹：./contributors
下，以自己的GitHub账号为名称，建立自己的py
文件夹，在其中进行创作。不要修改自己文件夹以
外的任何文件。
121 - - 例如：我需要给python-office添加一个
add方法。
122 - - 我的Github账户名
为：CoderWanFeng。
123 - - 于是我在./contributors新建了文
件夹./CoderWanFeng后，
124 - - 新建了add.py文件，编辑我的代码

117
118 python-office欢迎任何人来添砖加瓦，贡献代
码，建议提交的pr（pull request）符合一些规
范，规范如下：
119
120 + 参与项目建设的步骤：
121 + - 例如：你需要给python-office添加一个add方
法。
122 + 1. 你的Github账户名为：demo
123 + 2. 于是你在./contributors新建了文件夹。
/demo
124 + 3. 新建了add.py文件，编辑你的代码

125	1. 注释完备，尤其每个新增的方法应按照Google Python文档规范标明方法说明、参数说明、返回值说明等信息，必要时请添加单元测试，如果愿意，也可以加上你的大名。	125	+ 4. 编辑完成，提交pr到master分支（gitee或者GitHub，都可以）。可以注明你对自己功能的取名建议
126	- 2. python-office的文档，需要进行格式化。注意： <u>只需要格式化你自己的代码</u>	126	+ 5. 晚枫收到后，会对各位的代码进行测试后，合并后打包上传到python官方库
127	3. 请直接pull request到`master`分支。 `master`是主分支，表示已经发布pypi库的版本。 **未来参与人数增多，会开辟新的分支，请留意本文档的更新。**	127	+
128	4. 我们如果关闭了你的issue或pr，请不要诧异，这是我们保持问题处理整洁的一种方式，你依旧可以继续讨论，当有讨论结果时我们会重新打开。	128	+ ### 📄 代码规范
129		129	+
		130	1. 注释完备，尤其每个新增的方法应按照Google Python文档规范标明方法说明、参数说明、返回值说明等信息，必要时请添加单元测试，如果愿意，也可以加上你的大名。
		131	+ 2. python-office的文档，需要进行格式化。注意： <u>只能格式化你自己的代码</u>
		132	3. 请直接pull request到`master`分支。 `master`是主分支，表示已经发布pypi库的版本。 **未来参与人数增多，会开辟新的分支，请留意本文档的更新。**
		133	4. 我们如果关闭了你的issue或pr，请不要诧异，这是我们保持问题处理整洁的一种方式，你依旧可以继续讨论，当有讨论结果时我们会重新打开。
		134	

✓ 111 ■■■■■ contributors/ CNSeniorious000/pdf.py 📄

... @@ -0,0 +1,111 @@

```
1 + # from functools import cache,
    cached_property
2 + #
3 + #
4 + # # PDF与栅格化
5 + # # noinspection PyPackageRequirements
6 + # class PDF:
7 + #     """
8 + #     a pdf document with optimized lazy
    processing
9 + #     @Author & Date: CNSeniorious000
    2022/5/17
10 + #     """
11 + #
12 + #     @cache
13 + #     def __new__(cls, *args, **kwargs):
```

```
14 + #         """a path refers to only one
    + #         document"""
15 + #         return object.__new__(cls)
16 + #
17 + #     def __init__(self, path: str):
18 + #         """load from disk or
    + #         internet"""
19 + #         if path.startswith("http"):
20 + #             import requests
21 + #             self.raw =
    + #             requests.get(path).content
22 + #         else:
23 + #             self.raw = open(path,
    + #             "rb").read()
24 + #
25 + #     @cached_property
26 + #     def doc(self):
27 + #         import fitz
28 + #         # noinspection
    + #         PyUnresolvedReferences
29 + #         return
    + #         fitz.open(stream=self.raw)
30 + #
31 + #     @property
32 + #     def page_count(self):
33 + #         return self.doc.page_count
34 + #
35 + #     @cache
36 + #     def get_pixmap(self, page=0,
    + #     dpi=108, alpha=True):
37 + #         return
    + #         self.doc[page].get_pixmap(dpi=dpi,
    + #         alpha=alpha)
38 + #
39 + #     @cache
40 + #     def get_image(self, *args,
    + #     **kwargs):
41 + #         from imageio import imread
42 + #         from numpy import asarray
43 + #         return
    + #         asarray(imread(self.get_pixmap(*args,
    + #         **kwargs).tobytes()))
44 + #
```

```
45 + #         @staticmethod
46 + #         def show(image):
47 + #             from matplotlib.pyplot import
48 + #                 imshow, show
49 + #             imshow(image)
50 + #             return show()
51 + #
52 + #         def show_image(self, *args,
53 + #             **kwargs):
54 + #             return
55 + #                 self.show(self.get_image(*args,
56 + #                     **kwargs))
57 + #
58 + #         def save_image(self, file_path:
59 + #             str, page=0, dpi=144, alpha=True):
60 + #             return self.get_pixmap(page,
61 + #                 dpi, alpha).save(file_path)
62 + #
63 + #         def save_images(self, dir_path:
64 + #             str, pages=..., dpi=144, alpha=True,
65 + #                 encode="png", show_bar=True):
66 + #             if not
67 + #                 os.path.isdir(dir_path):
68 + #                     os.mkdir(dir_path)
69 + #
70 + #             it = range(self.page_count) if
71 + #                 pages is ... else pages
72 + #
73 + #             if show_bar:
74 + #                 from alive_progress import
75 + #                     alive_it
76 + #                 it = alive_it(it)
77 + #
78 + #             for page in it:
79 + #                 self.save_image(f"
80 + #                     {dir_path}/{page}.{encode}", page, dpi,
81 + #                         alpha)
82 + #
83 + #
84 + # # PDF与OCR
85 + # class PDFReader(PDF):
86 + #     def __init__(self, path: str,
87 + #         lang=("en", "ch_sim")):
```


```
74 + #         super().__init__(path)
75 + #         self.lang = lang
76 + #
77 + #     @cached_property
78 + #     def reader(self):
79 + #         from easyocr import Reader
80 + #         return Reader(self.lang)
81 + #
82 + #     @cache
83 + #     def get_texts(self, page=0, *args,
84 + #                  **kwargs):
85 + #         return
86 + #         self.reader.readtext(self.get_image(page
87 + #         , *args, **kwargs))
88 + #
89 + #     def render_bbox(self, page=0,
90 + #                     *args, color=(255, 0, 0), **kwargs):
91 + #         from cv2 import line
92 + #         image = self.get_image(page,
93 + #                                *args, **kwargs).copy()
94 + #         for points, string, degree in
95 + #             self.get_texts(page, *args, **kwargs):
96 + #             c = (*color, round(degree
97 + #                                * 255))
98 + #             # noinspection
99 + #             PyPep8Naming
100 + #             A, B, C, D =
101 + #                 [tuple(map(round, point)) for point in
102 + #                  points]
103 + #             line(image, A, B, c)
104 + #             line(image, B, C, c)
105 + #             line(image, C, D, c)
106 + #             line(image, D, A, c)
107 + #
108 + #         return image
109 + #
110 + #     def show_image_with_bbox(self,
111 + #                              *args, **kwargs):
112 + #         return
113 + #         self.show(self.render_bbox(*args,
114 + #                                     **kwargs))
115 + #
```

```

103 + #         def classify(self, choices,
104 + #             page=0, *args, **kwargs):
105 + #             """
106 + #             这个函数的功能在于，比如有一个奖
107 + #             状，上面可能有一等奖二等奖三等奖，要批量识别
108 + #             出一张图片为什么奖
109 + #             :param page: the page to ocr
110 + #             :param choices: list or words
111 + #             to choose
112 + #             """
113 + #             from rapidfuzz.process import
114 + #             extractOne
115 + #             texts = [string for _, string,
116 + #                 _ in self.get_texts(page, *args,
117 + #                     **kwargs)]
118 + #             return
119 + #             sorted((extractOne(choice, texts)[1],
120 + #                 choice) for choice in choices)[-1][1]

```


▼ 0 ■■■■■

service/image/eliminate_background.py → ...ors/CHENJie666666/eliminate_background.py 

File renamed without changes.

▼ 0 ■■■■■ contributors/CoderWanFeng/add.py → contributors/demo/demo.py 

File renamed without changes.

▼ ↕ 2 ■■■■■ office/image.py 

```

59
60     def add_watermark(file, mark,
61         out="output", color="#8B8B1B", size=30,
62         opacity=0.15, space=75, angle=30):
63         """
64         - @Author & Date : CoderWanFeng
65         2022/5/6 14:33
66         @Desc : 给图片添加水印
67         @Return : 添加了水印的图片，输出到out
68         指定的文件夹
69         """

```

```

59
60     def add_watermark(file, mark,
61         out="output", color="#8B8B1B", size=30,
62         opacity=0.15, space=75, angle=30):
63         """
64         + @Author & Date : demo 2022/5/6 14:33
65         @Desc : 给图片添加水印
66         @Return : 添加了水印的图片，输出到out
67         指定的文件夹
68         """

```

110 office/pdf.py

```

6     from PyPDF2 import PdfFileReader,
      PdfFileWriter
7     from pdf2docx import Converter
8
9     - # from functools import cache,
      cached_property
10
11     #给pdf加水印
12     def add_watermark():
13         cv.close()
14     except:
15         print('这个文件有问题~! ')
16
17     - #
18     - # # PDF与栅格化
19     - # # noinspection PyPackageRequirements
20     - # class PDF:
21     - #     """
22     - #     a pdf document with optimized lazy
23     processing
24     - #     @Author & Date: CNSeniorious000
25     2022/5/17
26     - #     """
27     - #
28     - #     @cache
29     - #     def __new__(cls, *args, **kwargs):
30     - #         """a path refers to only one
31     document"""
32     - #         return object.__new__(cls)
33     - #
34     - #     def __init__(self, path: str):
35     - #         """load from disk or
36     internet"""
37     - #         if path.startswith("http"):
38     - #             import requests
39     - #             self.raw =
40     requests.get(path).content
41     - #         else:
42     - #             self.raw = open(path,
43     "rb").read()
44     - #
45     - #     @cached_property

```

```

6     from PyPDF2 import PdfFileReader,
      PdfFileWriter
7     from pdf2docx import Converter
8
9
10    #给pdf加水印
11    def add_watermark():
12        cv.close()
13    except:
14        print('这个文件有问题~! ')

```

```
108 - #     def doc(self):
109 - #         import fitz
110 - #         # noinspection
111 - #         PyUnresolvedReferences
112 - #         return
113 - #         fitz.open(stream=self.raw)
114 - #
115 - #     @property
116 - #     def page_count(self):
117 - #         return self.doc.page_count
118 - #
119 - #     @cache
120 - #     def get_pixmap(self, page=0,
121 - #                    dpi=108, alpha=True):
122 - #         return
123 - #         self.doc[page].get_pixmap(dpi=dpi,
124 - #                                    alpha=alpha)
125 - #
126 - #     @cache
127 - #     def get_image(self, *args,
128 - #                  **kwargs):
129 - #         from imageio import imread
130 - #         from numpy import asarray
131 - #         return
132 - #         asarray(imread(self.get_pixmap(*args,
133 - #                                       **kwargs).tobytes()))
134 - #
135 - #     @staticmethod
136 - #     def show(image):
137 - #         from matplotlib.pyplot import
138 - #         imshow, show
139 - #         imshow(image)
140 - #         return show()
141 - #
142 - #     def show_image(self, *args,
143 - #                    **kwargs):
144 - #         return
145 - #         self.show(self.get_image(*args,
146 - #                                   **kwargs))
147 - #
148 - #     def save_image(self, file_path:
149 - #                   str, page=0, dpi=144, alpha=True):
```



```
137 - #         return self.get_pixmap(page,
138 - #         dpi, alpha).save(file_path)
139 - #
140 - #     def save_images(self, dir_path:
141 - #         str, pages=..., dpi=144, alpha=True,
142 - #         encode="png", show_bar=True):
143 - #         if not
144 - #             os.path.isdir(dir_path):
145 - #                 os.mkdir(dir_path)
146 - #
147 - #         it = range(self.page_count) if
148 - #             pages is ... else pages
149 - #
150 - #         if show_bar:
151 - #             from alive_progress import
152 - #                 alive_it
153 - #             it = alive_it(it)
154 - #
155 - #         for page in it:
156 - #             self.save_image(f"
157 - #                 {dir_path}/{page}.{encode}", page, dpi,
158 - #                 alpha)
159 - #
160 - #
161 - # # PDF与OCR
162 - # class PDFReader(PDF):
163 - #     def __init__(self, path: str,
164 - #         lang=("en", "ch_sim")):
165 - #         super().__init__(path)
166 - #         self.lang = lang
167 - #
168 - #     @cached_property
169 - #     def reader(self):
170 - #         from easyocr import Reader
171 - #         return Reader(self.lang)
172 - #
173 - #     @cache
174 - #     def get_texts(self, page=0, *args,
175 - #         **kwargs):
176 - #         return
177 - #             self.reader.readtext(self.get_image(page
178 - #                 , *args, **kwargs))
179 - #
```

```
168 - #     def render_bbox(self, page=0,
      - #         args, color=(255, 0, 0), **kwargs):
169 - #         from cv2 import line
170 - #         image = self.get_image(page,
      - #             args, **kwargs).copy()
171 - #         for points, string, degree in
      - #             self.get_texts(page, *args, **kwargs):
172 - #             c = (*color, round(degree
      - #                 * 255))
173 - #             # noinspection
      - #                 PyPep8Naming
174 - #             A, B, C, D =
      - #                 [tuple(map(round, point)) for point in
      - #                     points]
175 - #             line(image, A, B, c)
176 - #             line(image, B, C, c)
177 - #             line(image, C, D, c)
178 - #             line(image, D, A, c)
179 - #
180 - #             return image
181 - #
182 - #     def show_image_with_bbox(self,
      - #         args, **kwargs):
183 - #         return
      - #             self.show(self.render_bbox(*args,
      - #                 **kwargs))
184 - #
185 - #     def classify(self, choices,
      - #         page=0, *args, **kwargs):
186 - #         """
187 - #         这个函数的功能在于，比如有一个奖
      - #             状，上面可能有一等奖二等奖三等奖，要批量识别
      - #             出一张图片为什么奖
188 - #         :param page: the page to ocr
189 - #         :param choices: list or words
      - #             to choose
190 - #         """
191 - #         from rapidfuzz.process import
      - #             extractOne
192 - #         texts = [string for _, string,
      - #             _ in self.get_texts(page, *args,
      - #                 **kwargs)]
```

```
193 - #         return
      sorted((extractOne(choice, texts)[1],
             choice) for choice in choices)[-1][1]
```

0 comments on commit [7217772](#)