

# vimのカラースキーム 設定 + 編集 について

# 自己紹介

- 氏名：星野 華
- 学年：学部4年
- 所属：横田理央研究室
- vim歴：2.5年
- 趣味：
  - カラースキームサーフィング
  - カラースキーム編集 + 作成
  - ターミナルのビジュアル強化
- Github: **sff1019**

# How I spend my time:



# 0. カラースキームとは

## 0.1 カラースキームとは

vimの編集画面に色をつけてくれるもの

## 0.1 カラースキームとは

“

vimを使いたくさせるもの

”

# カラースキームがないと…

```
16 learning_rate = 0.001
17 batch_size = 50
18 step_size = 3
19
20 # Fully connected neural network with one hidden layer
21
22
23 class NeuralNet(nn.Module):
24     def __init__(self, input_size, hidden_size):
25         super(NeuralNet, self).__init__()
26         self.fc1 = nn.Linear(input_size, hidden_size * 2)
27         self.relu1 = nn.ReLU()
28         self.fc2 = nn.Linear(hidden_size * 2, hidden_size)
29         self.relu2 = nn.ReLU()
30         self.fc3 = nn.Linear(hidden_size, 1)
31
32     def forward(self, x):
33         out = self.fc1(x)
34         out = self.relu1(out)
35         out = self.fc2(out)
36         out = self.relu2(out)
37         out = self.fc3(out)
38         return out
39
40
41 model = NeuralNet(input_size, hidden_size).to(device)
42
43 # Loss and optimizer
44 criterion = nn.MSELoss()
45 optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
46
47 train_dataset = list(range(0, step_size))
48 # val_dataset = list(range(0, 26))
49
50 print('start training')
51 # Train the model
52 for epoch in range(num_epochs):
53     for i in train_dataset:
54         # Move tensors to the configured device
55         filename = 'models/preprocessed_{}.pickle'.format(i)
56         with open(filename, 'rb') as f:
57             print('loading data')
58             train_data = pickle.load(f).astype(np.float32)
59             print('finish_loading')
60
```

見えづらい・何が何だかわかりづらい・かっこ悪い

# カラースキームを追加すると…

```
16 learning_rate = 0.001
17 batch_size = 50
18 step_size = 3
19
20 # Fully connected neural network with one hidden layer
21
22
23 class NeuralNet(nn.Module):
24     def __init__(self, input_size, hidden_size):
25         super(NeuralNet, self).__init__()
26         self.fc1 = nn.Linear(input_size, hidden_size * 2)
27         self.relu1 = nn.ReLU()
28         self.fc2 = nn.Linear(hidden_size * 2, hidden_size)
29         self.relu2 = nn.ReLU()
30         self.fc3 = nn.Linear(hidden_size, 1)
31
32     def forward(self, x):
33         out = self.fc1(x)
34         out = self.relu1(out)
35         out = self.fc2(out)
36         out = self.relu2(out)
37         out = self.fc3(out)
38         return out
39
40
41 model = NeuralNet(input_size, hidden_size).to(device)
42
43 # Loss and optimizer
44 criterion = nn.MSELoss()
45 optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)
46
47 train_dataset = list(range(0, step_size))
48 # val_dataset = list(range(0, 26))
49
50 print('start training')
51 # Train the model
52 for epoch in range(num_epochs):
53     for i in train_dataset:
54         # Move tensors to the configured device
55         filename = 'models/preprocessed_{}.pickle'.format(i)
56         with open(filename, 'rb') as f:
57             print('loading data')
58             train_data = pickle.load(f).astype(np.float32)
59             print('finish_loading')
60
```

見えやすい！・シンタックスがわかりやすい・かっこいい！

```
vim 10.0 %
format.py ✘ format_individual.py ✘ eval.py ✘ preprocess.py ✘ submit.py ✘ word2vec.py ✘
1 import csv
2 import sys
3 import re
4 import time
5 import pickle
6 from itertools import chain
7
8 import numpy as np
9
10
11 if __name__ == '__main__':
12
13     path = 'mercari/test_stg2.tsv'
14     # path = 'mercari/train.tsv'
15     f = open(path, 'r', encoding='utf-8')
16     tsv = csv.reader(f, delimiter='\t')
17     next(tsv)
18
19     with open('processed/ohe_item_condition.pkl', 'rb') as f:
20         ohe_condition = pickle.load(f)
21
22     with open('processed/le_item_condition.pkl', 'rb') as f:
23         le_condition = pickle.load(f)
24
25     with open('processed/word2vec_name.pkl', 'rb') as f:
26         word2vec_name = pickle.load(f)
27
28     with open('processed/word2vec_item_desc.pkl', 'rb') as f:
29         word2vec_item_description = pickle.load(f)
30
31     with open('processed/word2vec_brand_name.pkl', 'rb') as f:
32         word2vec_brand_name = pickle.load(f)
33
34     with open('processed/word2vec_category_name.pkl', 'rb') as f:
35         word2vec_category_name = pickle.load(f)
36
37     print('Reading...')
38     sys.stdout.flush()
39     rows = []
40     start_time = time.time()
41     count = 0
42     for index, line in enumerate(tsv):
43         condition = le_condition.transform([line[2]]).reshape(1, 1)
44         np_condition = ohe_condition.transform(condition).flatten()
45
46         name = [s for s in [s.strip().lower().replace('/', ' ') for s in re.split('.+', line[1])]]
47         name = list(chain.from_iterable(name))
48         np_name = word2vec_name.wv[name].mean(axis=0).flatten()
49
50         # desc = [s for s in [s.strip().lower().replace('/', ' ') for s in re.split('.+', line[7])]]
51         # np_desc = word2vec_name.wv[name].mean(axis=0).flatten()
52
53
```

```
vim 10.0 %
format.py ✘ format_individual.py ✘ eval.py ✘ preprocess.py ✘ submit.py ✘ word2vec.py ✘
1 import csv
2 import sys
3 import re
4 import time
5 import pickle
6 from itertools import chain
7
8 import numpy as np
9
10
11 if __name__ == '__main__':
12
13     path = 'mercari/test_stg2.tsv'
14
15     with open('processed/ohe_item_condition.pkl', 'rb') as f:
16         ohe_condition = pickle.load(f)
17
18     with open('processed/le_item_condition.pkl', 'rb') as f:
19         le_condition = pickle.load(f)
20
21     with open('processed/word2vec_name.pkl', 'rb') as f:
22         word2vec_name = pickle.load(f)
23
24     with open('processed/word2vec_item_desc.pkl', 'rb') as f:
25         word2vec_item_description = pickle.load(f)
26
27     with open('processed/word2vec_brand_name.pkl', 'rb') as f:
28         word2vec_brand_name = pickle.load(f)
29
30     with open('processed/word2vec_category_name.pkl', 'rb') as f:
31         word2vec_category_name = pickle.load(f)
32
33     print('Reading...')
34     sys.stdout.flush()
35     rows = []
36     start_time = time.time()
37     count = 0
38     for index, line in enumerate(tsv):
39         condition = le_condition.transform([line[2]]).reshape(1, 1)
40         np_condition = ohe_condition.transform(condition).flatten()
41
42         name = [s for s in [s.strip().lower().replace('/', ' ') for s in re.split('.+', line[1])]]
43         name = list(chain.from_iterable(name))
44         np_name = word2vec_name.wv[name].mean(axis=0).flatten()
45
46         # desc = [s for s in [s.strip().lower().replace('/', ' ') for s in re.split('.+', line[7])]]
47         # np_desc = word2vec_name.wv[name].mean(axis=0).flatten()
48
49
50
51
52
53
```

# カラースキームは偉大！



# もちろん種類はたくさんあるのが vimの良いところ！

```
1 // Create a Liquid-underlable version of this Document
2 defaults = &site.frontmatter.defaults.all[url, collection.label.to_sym]
3 unless defaults.empty?
4   defaults = Multi::Frontmatter::Merge::Object.new(defaults)
5   defaults.merge!(site.content)
6   site.content = File.read(path, merged_file_read_opts(opts))
7   if content =~ YAML_FRONT_MATTER_REGEX
8     YAML::safe_load(content)
9     content = YAML::safe_dump($1)
10    unless data_file.nil?
11      data_file.write(data)
12    end
13  end
14 end
15 rescue SyntaxError =>
16   puts "YAML Exception reading #{$path}: #{$e.message}"
17   raise "Error reading file #{$path}: #{$e.message}"
18 end
19
20 # Create a Liquid-underlable version of this Document
21 # Returns a Hash representing this Document's data.
22 # Options:
23 #   - &url (String) -> url
24 #   - &collection (String) -> collection
25 #   - &label (String) -> collection.label
26
27 let s:save_cpo = &cpoptions
28 set cpoptions&vim
29
30 function! s:keep_buffer_singularity() abort
31   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
32   if len(related_win_ids) <= 1
33     return 1
34   endif
35   let s:save_cpo = &cpoptions
36   set cpoptions&vim
37
38   call vaffle#buffer#duplicate()
39
40   return 1
41
42 function! s:get_cursor_items(env, mode) abort
43   let items = a:env.items
44   if empty(items)
45     return []
46   endif
47
48   " Detected multiple windows for single buffer:
49   " Duplicate the buffer to avoid unwanted sync between different windows
50   call vaffle#buffer#duplicate()
51
52   return 1
53
54 endfunction
55
56 function! s:get_cursor_items(env, mode) abort
57   let items = a:env.items
58   if empty(items)
59     return []
60   endif
61
62   " Detected multiple windows for single buffer:
63   " Duplicate the buffer to avoid unwanted sync between different windows
64   call vaffle#buffer#duplicate()
65
66   return 1
67
68 endfunction
69
70 function! s:keep_buffer_singularity() abort
71   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
72   if len(related_win_ids) <= 1
73     return 1
74   endif
75   let s:save_cpo = &cpoptions
76   set cpoptions&vim
77
78   call vaffle#buffer#duplicate()
79
80   return 1
81
82 endfunction
83
84 function! s:keep_buffer_singularity() abort
85   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
86   if len(related_win_ids) <= 1
87     return 1
88   endif
89   let s:save_cpo = &cpoptions
90   set cpoptions&vim
91
92   call vaffle#buffer#duplicate()
93
94   return 1
95
96 endfunction
97
98 function! s:keep_buffer_singularity() abort
99   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
100  if len(related_win_ids) <= 1
101    return 1
102  endif
103  let s:save_cpo = &cpoptions
104  set cpoptions&vim
105
106  call vaffle#buffer#duplicate()
107
108  return 1
109
110 endfunction
111
112 function! s:keep_buffer_singularity() abort
113   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
114   if len(related_win_ids) <= 1
115     return 1
116   endif
117   let s:save_cpo = &cpoptions
118   set cpoptions&vim
119
120   call vaffle#buffer#duplicate()
121
122   return 1
123
124 endfunction
125
126 function! s:keep_buffer_singularity() abort
127   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
128   if len(related_win_ids) <= 1
129     return 1
130   endif
131   let s:save_cpo = &cpoptions
132   set cpoptions&vim
133
134   call vaffle#buffer#duplicate()
135
136   return 1
137
138 endfunction
139
140 function! s:keep_buffer_singularity() abort
141   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
142   if len(related_win_ids) <= 1
143     return 1
144   endif
145   let s:save_cpo = &cpoptions
146   set cpoptions&vim
147
148   call vaffle#buffer#duplicate()
149
150   return 1
151
152 endfunction
153
154 function! s:keep_buffer_singularity() abort
155   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
156   if len(related_win_ids) <= 1
157     return 1
158   endif
159   let s:save_cpo = &cpoptions
160   set cpoptions&vim
161
162   call vaffle#buffer#duplicate()
163
164   return 1
165
166 endfunction
167
168 function! s:keep_buffer_singularity() abort
169   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
170   if len(related_win_ids) <= 1
171     return 1
172   endif
173   let s:save_cpo = &cpoptions
174   set cpoptions&vim
175
176   call vaffle#buffer#duplicate()
177
178   return 1
179
180 endfunction
181
182 function! s:keep_buffer_singularity() abort
183   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
184   if len(related_win_ids) <= 1
185     return 1
186   endif
187   let s:save_cpo = &cpoptions
188   set cpoptions&vim
189
190   call vaffle#buffer#duplicate()
191
192   return 1
193
194 endfunction
195
196 function! s:keep_buffer_singularity() abort
197   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
198   if len(related_win_ids) <= 1
199     return 1
200   endif
201   let s:save_cpo = &cpoptions
202   set cpoptions&vim
203
204   call vaffle#buffer#duplicate()
205
206   return 1
207
208 endfunction
209
210 function! s:keep_buffer_singularity() abort
211   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
212   if len(related_win_ids) <= 1
213     return 1
214   endif
215   let s:save_cpo = &cpoptions
216   set cpoptions&vim
217
218   call vaffle#buffer#duplicate()
219
220   return 1
221
222 endfunction
223
224 function! s:keep_buffer_singularity() abort
225   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
226   if len(related_win_ids) <= 1
227     return 1
228   endif
229   let s:save_cpo = &cpoptions
230   set cpoptions&vim
231
232   call vaffle#buffer#duplicate()
233
234   return 1
235
236 endfunction
237
238 function! s:keep_buffer_singularity() abort
239   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
240   if len(related_win_ids) <= 1
241     return 1
242   endif
243   let s:save_cpo = &cpoptions
244   set cpoptions&vim
245
246   call vaffle#buffer#duplicate()
247
248   return 1
249
250 endfunction
251
252 function! s:keep_buffer_singularity() abort
253   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
254   if len(related_win_ids) <= 1
255     return 1
256   endif
257   let s:save_cpo = &cpoptions
258   set cpoptions&vim
259
260   call vaffle#buffer#duplicate()
261
262   return 1
263
264 endfunction
265
266 function! s:keep_buffer_singularity() abort
267   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
268   if len(related_win_ids) <= 1
269     return 1
270   endif
271   let s:save_cpo = &cpoptions
272   set cpoptions&vim
273
274   call vaffle#buffer#duplicate()
275
276   return 1
277
278 endfunction
279
280 function! s:keep_buffer_singularity() abort
281   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
282   if len(related_win_ids) <= 1
283     return 1
284   endif
285   let s:save_cpo = &cpoptions
286   set cpoptions&vim
287
288   call vaffle#buffer#duplicate()
289
290   return 1
291
292 endfunction
293
294 function! s:keep_buffer_singularity() abort
295   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
296   if len(related_win_ids) <= 1
297     return 1
298   endif
299   let s:save_cpo = &cpoptions
300   set cpoptions&vim
301
302   call vaffle#buffer#duplicate()
303
304   return 1
305
306 endfunction
307
308 function! s:keep_buffer_singularity() abort
309   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
310   if len(related_win_ids) <= 1
311     return 1
312   endif
313   let s:save_cpo = &cpoptions
314   set cpoptions&vim
315
316   call vaffle#buffer#duplicate()
317
318   return 1
319
320 endfunction
321
322 function! s:keep_buffer_singularity() abort
323   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
324   if len(related_win_ids) <= 1
325     return 1
326   endif
327   let s:save_cpo = &cpoptions
328   set cpoptions&vim
329
330   call vaffle#buffer#duplicate()
331
332   return 1
333
334 endfunction
335
336 function! s:keep_buffer_singularity() abort
337   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
338   if len(related_win_ids) <= 1
339     return 1
340   endif
341   let s:save_cpo = &cpoptions
342   set cpoptions&vim
343
344   call vaffle#buffer#duplicate()
345
346   return 1
347
348 endfunction
349
350 function! s:keep_buffer_singularity() abort
351   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
352   if len(related_win_ids) <= 1
353     return 1
354   endif
355   let s:save_cpo = &cpoptions
356   set cpoptions&vim
357
358   call vaffle#buffer#duplicate()
359
360   return 1
361
362 endfunction
363
364 function! s:keep_buffer_singularity() abort
365   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
366   if len(related_win_ids) <= 1
367     return 1
368   endif
369   let s:save_cpo = &cpoptions
370   set cpoptions&vim
371
372   call vaffle#buffer#duplicate()
373
374   return 1
375
376 endfunction
377
378 function! s:keep_buffer_singularity() abort
379   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
380   if len(related_win_ids) <= 1
381     return 1
382   endif
383   let s:save_cpo = &cpoptions
384   set cpoptions&vim
385
386   call vaffle#buffer#duplicate()
387
388   return 1
389
390 endfunction
391
392 function! s:keep_buffer_singularity() abort
393   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
394   if len(related_win_ids) <= 1
395     return 1
396   endif
397   let s:save_cpo = &cpoptions
398   set cpoptions&vim
399
400   call vaffle#buffer#duplicate()
401
402   return 1
403
404 endfunction
405
406 function! s:keep_buffer_singularity() abort
407   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
408   if len(related_win_ids) <= 1
409     return 1
410   endif
411   let s:save_cpo = &cpoptions
412   set cpoptions&vim
413
414   call vaffle#buffer#duplicate()
415
416   return 1
417
418 endfunction
419
420 function! s:keep_buffer_singularity() abort
421   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
422   if len(related_win_ids) <= 1
423     return 1
424   endif
425   let s:save_cpo = &cpoptions
426   set cpoptions&vim
427
428   call vaffle#buffer#duplicate()
429
430   return 1
431
432 endfunction
433
434 function! s:keep_buffer_singularity() abort
435   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
436   if len(related_win_ids) <= 1
437     return 1
438   endif
439   let s:save_cpo = &cpoptions
440   set cpoptions&vim
441
442   call vaffle#buffer#duplicate()
443
444   return 1
445
446 endfunction
447
448 function! s:keep_buffer_singularity() abort
449   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
450   if len(related_win_ids) <= 1
451     return 1
452   endif
453   let s:save_cpo = &cpoptions
454   set cpoptions&vim
455
456   call vaffle#buffer#duplicate()
457
458   return 1
459
460 endfunction
461
462 function! s:keep_buffer_singularity() abort
463   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
464   if len(related_win_ids) <= 1
465     return 1
466   endif
467   let s:save_cpo = &cpoptions
468   set cpoptions&vim
469
470   call vaffle#buffer#duplicate()
471
472   return 1
473
474 endfunction
475
476 function! s:keep_buffer_singularity() abort
477   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
478   if len(related_win_ids) <= 1
479     return 1
480   endif
481   let s:save_cpo = &cpoptions
482   set cpoptions&vim
483
484   call vaffle#buffer#duplicate()
485
486   return 1
487
488 endfunction
489
490 function! s:keep_buffer_singularity() abort
491   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
492   if len(related_win_ids) <= 1
493     return 1
494   endif
495   let s:save_cpo = &cpoptions
496   set cpoptions&vim
497
498   call vaffle#buffer#duplicate()
499
500   return 1
501
502 endfunction
503
504 function! s:keep_buffer_singularity() abort
505   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
506   if len(related_win_ids) <= 1
507     return 1
508   endif
509   let s:save_cpo = &cpoptions
510   set cpoptions&vim
511
512   call vaffle#buffer#duplicate()
513
514   return 1
515
516 endfunction
517
518 function! s:keep_buffer_singularity() abort
519   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
520   if len(related_win_ids) <= 1
521     return 1
522   endif
523   let s:save_cpo = &cpoptions
524   set cpoptions&vim
525
526   call vaffle#buffer#duplicate()
527
528   return 1
529
530 endfunction
531
532 function! s:keep_buffer_singularity() abort
533   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
534   if len(related_win_ids) <= 1
535     return 1
536   endif
537   let s:save_cpo = &cpoptions
538   set cpoptions&vim
539
540   call vaffle#buffer#duplicate()
541
542   return 1
543
544 endfunction
545
546 function! s:keep_buffer_singularity() abort
547   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
548   if len(related_win_ids) <= 1
549     return 1
550   endif
551   let s:save_cpo = &cpoptions
552   set cpoptions&vim
553
554   call vaffle#buffer#duplicate()
555
556   return 1
557
558 endfunction
559
560 function! s:keep_buffer_singularity() abort
561   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
562   if len(related_win_ids) <= 1
563     return 1
564   endif
565   let s:save_cpo = &cpoptions
566   set cpoptions&vim
567
568   call vaffle#buffer#duplicate()
569
570   return 1
571
572 endfunction
573
574 function! s:keep_buffer_singularity() abort
575   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
576   if len(related_win_ids) <= 1
577     return 1
578   endif
579   let s:save_cpo = &cpoptions
580   set cpoptions&vim
581
582   call vaffle#buffer#duplicate()
583
584   return 1
585
586 endfunction
587
588 function! s:keep_buffer_singularity() abort
589   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
590   if len(related_win_ids) <= 1
591     return 1
592   endif
593   let s:save_cpo = &cpoptions
594   set cpoptions&vim
595
596   call vaffle#buffer#duplicate()
597
598   return 1
599
600 endfunction
601
602 function! s:keep_buffer_singularity() abort
603   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
604   if len(related_win_ids) <= 1
605     return 1
606   endif
607   let s:save_cpo = &cpoptions
608   set cpoptions&vim
609
610   call vaffle#buffer#duplicate()
611
612   return 1
613
614 endfunction
615
616 function! s:keep_buffer_singularity() abort
617   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
618   if len(related_win_ids) <= 1
619     return 1
620   endif
621   let s:save_cpo = &cpoptions
622   set cpoptions&vim
623
624   call vaffle#buffer#duplicate()
625
626   return 1
627
628 endfunction
629
630 function! s:keep_buffer_singularity() abort
631   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
632   if len(related_win_ids) <= 1
633     return 1
634   endif
635   let s:save_cpo = &cpoptions
636   set cpoptions&vim
637
638   call vaffle#buffer#duplicate()
639
640   return 1
641
642 endfunction
643
644 function! s:keep_buffer_singularity() abort
645   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
646   if len(related_win_ids) <= 1
647     return 1
648   endif
649   let s:save_cpo = &cpoptions
650   set cpoptions&vim
651
652   call vaffle#buffer#duplicate()
653
654   return 1
655
656 endfunction
657
658 function! s:keep_buffer_singularity() abort
659   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
660   if len(related_win_ids) <= 1
661     return 1
662   endif
663   let s:save_cpo = &cpoptions
664   set cpoptions&vim
665
666   call vaffle#buffer#duplicate()
667
668   return 1
669
670 endfunction
671
672 function! s:keep_buffer_singularity() abort
673   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
674   if len(related_win_ids) <= 1
675     return 1
676   endif
677   let s:save_cpo = &cpoptions
678   set cpoptions&vim
679
680   call vaffle#buffer#duplicate()
681
682   return 1
683
684 endfunction
685
686 function! s:keep_buffer_singularity() abort
687   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
688   if len(related_win_ids) <= 1
689     return 1
690   endif
691   let s:save_cpo = &cpoptions
692   set cpoptions&vim
693
694   call vaffle#buffer#duplicate()
695
696   return 1
697
698 endfunction
699
700 function! s:keep_buffer_singularity() abort
701   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
702   if len(related_win_ids) <= 1
703     return 1
704   endif
705   let s:save_cpo = &cpoptions
706   set cpoptions&vim
707
708   call vaffle#buffer#duplicate()
709
710   return 1
711
712 endfunction
713
714 function! s:keep_buffer_singularity() abort
715   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
716   if len(related_win_ids) <= 1
717     return 1
718   endif
719   let s:save_cpo = &cpoptions
720   set cpoptions&vim
721
722   call vaffle#buffer#duplicate()
723
724   return 1
725
726 endfunction
727
728 function! s:keep_buffer_singularity() abort
729   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
730   if len(related_win_ids) <= 1
731     return 1
732   endif
733   let s:save_cpo = &cpoptions
734   set cpoptions&vim
735
736   call vaffle#buffer#duplicate()
737
738   return 1
739
740 endfunction
741
742 function! s:keep_buffer_singularity() abort
743   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
744   if len(related_win_ids) <= 1
745     return 1
746   endif
747   let s:save_cpo = &cpoptions
748   set cpoptions&vim
749
750   call vaffle#buffer#duplicate()
751
752   return 1
753
754 endfunction
755
756 function! s:keep_buffer_singularity() abort
757   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
758   if len(related_win_ids) <= 1
759     return 1
760   endif
761   let s:save_cpo = &cpoptions
762   set cpoptions&vim
763
764   call vaffle#buffer#duplicate()
765
766   return 1
767
768 endfunction
769
770 function! s:keep_buffer_singularity() abort
771   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
772   if len(related_win_ids) <= 1
773     return 1
774   endif
775   let s:save_cpo = &cpoptions
776   set cpoptions&vim
777
778   call vaffle#buffer#duplicate()
779
780   return 1
781
782 endfunction
783
784 function! s:keep_buffer_singularity() abort
785   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
786   if len(related_win_ids) <= 1
787     return 1
788   endif
789   let s:save_cpo = &cpoptions
790   set cpoptions&vim
791
792   call vaffle#buffer#duplicate()
793
794   return 1
795
796 endfunction
797
798 function! s:keep_buffer_singularity() abort
799   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
800   if len(related_win_ids) <= 1
801     return 1
802   endif
803   let s:save_cpo = &cpoptions
804   set cpoptions&vim
805
806   call vaffle#buffer#duplicate()
807
808   return 1
809
810 endfunction
811
812 function! s:keep_buffer_singularity() abort
813   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
814   if len(related_win_ids) <= 1
815     return 1
816   endif
817   let s:save_cpo = &cpoptions
818   set cpoptions&vim
819
820   call vaffle#buffer#duplicate()
821
822   return 1
823
824 endfunction
825
826 function! s:keep_buffer_singularity() abort
827   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
828   if len(related_win_ids) <= 1
829     return 1
830   endif
831   let s:save_cpo = &cpoptions
832   set cpoptions&vim
833
834   call vaffle#buffer#duplicate()
835
836   return 1
837
838 endfunction
839
840 function! s:keep_buffer_singularity() abort
841   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
842   if len(related_win_ids) <= 1
843     return 1
844   endif
845   let s:save_cpo = &cpoptions
846   set cpoptions&vim
847
848   call vaffle#buffer#duplicate()
849
850   return 1
851
852 endfunction
853
854 function! s:keep_buffer_singularity() abort
855   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
856   if len(related_win_ids) <= 1
857     return 1
858   endif
859   let s:save_cpo = &cpoptions
860   set cpoptions&vim
861
862   call vaffle#buffer#duplicate()
863
864   return 1
865
866 endfunction
867
868 function! s:keep_buffer_singularity() abort
869   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
870   if len(related_win_ids) <= 1
871     return 1
872   endif
873   let s:save_cpo = &cpoptions
874   set cpoptions&vim
875
876   call vaffle#buffer#duplicate()
877
878   return 1
879
880 endfunction
881
882 function! s:keep_buffer_singularity() abort
883   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
884   if len(related_win_ids) <= 1
885     return 1
886   endif
887   let s:save_cpo = &cpoptions
888   set cpoptions&vim
889
890   call vaffle#buffer#duplicate()
891
892   return 1
893
894 endfunction
895
896 function! s:keep_buffer_singularity() abort
897   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
898   if len(related_win_ids) <= 1
899     return 1
900   endif
901   let s:save_cpo = &cpoptions
902   set cpoptions&vim
903
904   call vaffle#buffer#duplicate()
905
906   return 1
907
908 endfunction
909
910 function! s:keep_buffer_singularity() abort
911   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
912   if len(related_win_ids) <= 1
913     return 1
914   endif
915   let s:save_cpo = &cpoptions
916   set cpoptions&vim
917
918   call vaffle#buffer#duplicate()
919
920   return 1
921
922 endfunction
923
924 function! s:keep_buffer_singularity() abort
925   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
926   if len(related_win_ids) <= 1
927     return 1
928   endif
929   let s:save_cpo = &cpoptions
930   set cpoptions&vim
931
932   call vaffle#buffer#duplicate()
933
934   return 1
935
936 endfunction
937
938 function! s:keep_buffer_singularity() abort
939   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
940   if len(related_win_ids) <= 1
941     return 1
942   endif
943   let s:save_cpo = &cpoptions
944   set cpoptions&vim
945
946   call vaffle#buffer#duplicate()
947
948   return 1
949
950 endfunction
951
952 function! s:keep_buffer_singularity() abort
953   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
954   if len(related_win_ids) <= 1
955     return 1
956   endif
957   let s:save_cpo = &cpoptions
958   set cpoptions&vim
959
960   call vaffle#buffer#duplicate()
961
962   return 1
963
964 endfunction
965
966 function! s:keep_buffer_singularity() abort
967   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
968   if len(related_win_ids) <= 1
969     return 1
970   endif
971   let s:save_cpo = &cpoptions
972   set cpoptions&vim
973
974   call vaffle#buffer#duplicate()
975
976   return 1
977
978 endfunction
979
980 function! s:keep_buffer_singularity() abort
981   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
982   if len(related_win_ids) <= 1
983     return 1
984   endif
985   let s:save_cpo = &cpoptions
986   set cpoptions&vim
987
988   call vaffle#buffer#duplicate()
989
990   return 1
991
992 endfunction
993
994 function! s:keep_buffer_singularity() abort
995   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
996   if len(related_win_ids) <= 1
997     return 1
998   endif
999   let s:save_cpo = &cpoptions
1000  set cpoptions&vim
1001
1002  call vaffle#buffer#duplicate()
1003
1004  return 1
1005
1006 endfunction
1007
1008 function! s:keep_buffer_singularity() abort
1009   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1010  if len(related_win_ids) <= 1
1011    return 1
1012  endif
1013  let s:save_cpo = &cpoptions
1014  set cpoptions&vim
1015
1016  call vaffle#buffer#duplicate()
1017
1018  return 1
1019
1020 endfunction
1021
1022 function! s:keep_buffer_singularity() abort
1023   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1024   if len(related_win_ids) <= 1
1025     return 1
1026   endif
1027   let s:save_cpo = &cpoptions
1028   set cpoptions&vim
1029
1030   call vaffle#buffer#duplicate()
1031
1032   return 1
1033
1034 endfunction
1035
1036 function! s:keep_buffer_singularity() abort
1037   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1038   if len(related_win_ids) <= 1
1039     return 1
1040   endif
1041   let s:save_cpo = &cpoptions
1042   set cpoptions&vim
1043
1044   call vaffle#buffer#duplicate()
1045
1046   return 1
1047
1048 endfunction
1049
1050 function! s:keep_buffer_singularity() abort
1051   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1052   if len(related_win_ids) <= 1
1053     return 1
1054   endif
1055   let s:save_cpo = &cpoptions
1056   set cpoptions&vim
1057
1058   call vaffle#buffer#duplicate()
1059
1060   return 1
1061
1062 endfunction
1063
1064 function! s:keep_buffer_singularity() abort
1065   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1066   if len(related_win_ids) <= 1
1067     return 1
1068   endif
1069   let s:save_cpo = &cpoptions
1070   set cpoptions&vim
1071
1072   call vaffle#buffer#duplicate()
1073
1074   return 1
1075
1076 endfunction
1077
1078 function! s:keep_buffer_singularity() abort
1079   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1080   if len(related_win_ids) <= 1
1081     return 1
1082   endif
1083   let s:save_cpo = &cpoptions
1084   set cpoptions&vim
1085
1086   call vaffle#buffer#duplicate()
1087
1088   return 1
1089
1090 endfunction
1091
1092 function! s:keep_buffer_singularity() abort
1093   let related_win_ids = vaffle#compat#win_findbuf(bufnr('%'))
1094   if len(related_win_ids) <= 1
1095     return 1
1096   endif
1097   let s:save_cpo = &cpoptions
1098   set cpoptions&vim
1099
1100  call vaffle#buffer#duplicate()
1101
1102  return 1
1103
1104 endfunction
1105
1106 function! s:keep_buffer_singularity() abort
1107   let related_win_ids = vaffle#compat#win_findbuf(buf
```

# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用著作権等に注意
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. 自作カラースキームを公開する



レベル1  
デフォルトのカラースキーム



# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. 自作カラースキームを公開する

# その前にコマンド！

vim上でカラースキームを変更することができます！  
(一時的ではあるが)

`:coloscheme [new_colorscheme]`

または

`:colo [new_colorscheme]`

# 豆知識 ①

vim上で現在あるカラースキームの種類がわかる！

:coloSPACETAB

または

:coloSPACETAB

```
~ blue brogrammer darkblue default delek desert elford evening industry koehler morning murphy pablo peachpuff ron shine slate solarized torte zellner
:colo blue
```

全体に反映したい！

.vimrcに記述すればいいだけ！

# .vimrcってなんだよ

## .vimrcとは

vimの設定を記述する

専用ファイル

rc: run commands

の略という説がある

## .vimrcの探し方

`echo $MYVIMRC`

とすると、

vimが参照している

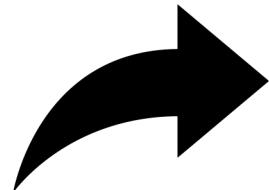
.vimrcが得られる



# で、どう記述するの？

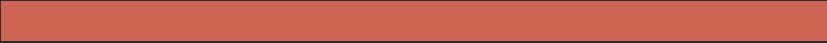
例えば。  
blueにしたい時

```
syntax enable  
colorscheme blue
```



再起動するところなるはず！

```
2 " Fastest way to move buffer  
3 nnoremap <silent><Left> :bp<CR>  
4 nnoremap <silent><Right> :bn<CR>  
5 nnoremap <silent><C-Space> :call BufferDeleteExceptFiler( )<CR>  
6  
7 function! BufferDeleteExceptFiler()  
8   if (&filetype !=# 'vimfiler')  
9     bd!  
0   endif  
1 endfunction  
2  
3 if !has('gui_running')  
4   augroup term_vim_c_space  
5     autocmd!  
6     autocmd VimEnter * map <Nul> <C-Space>  
7     autocmd VimEnter * map! <Nul> <C-Space>  
8   augroup END  
9 endif  
0  
1 " Display another buffer when current buffer isn't saved.  
2 set hidden  
3  
4 " Do not create swap files  
5 set noswapfile  
6  
7 " Set clipboard  
8 set clipboard+=%unnamedplus  
9  
0 " Set background dark
```



# レベル2 公開されたカラースキーム



# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. 自作カラースキームを公開する

# 公開されたカラースキームとは？

Githubなどに公開されたカラースキーム

molokai

gruvbox

solarized

hybrid

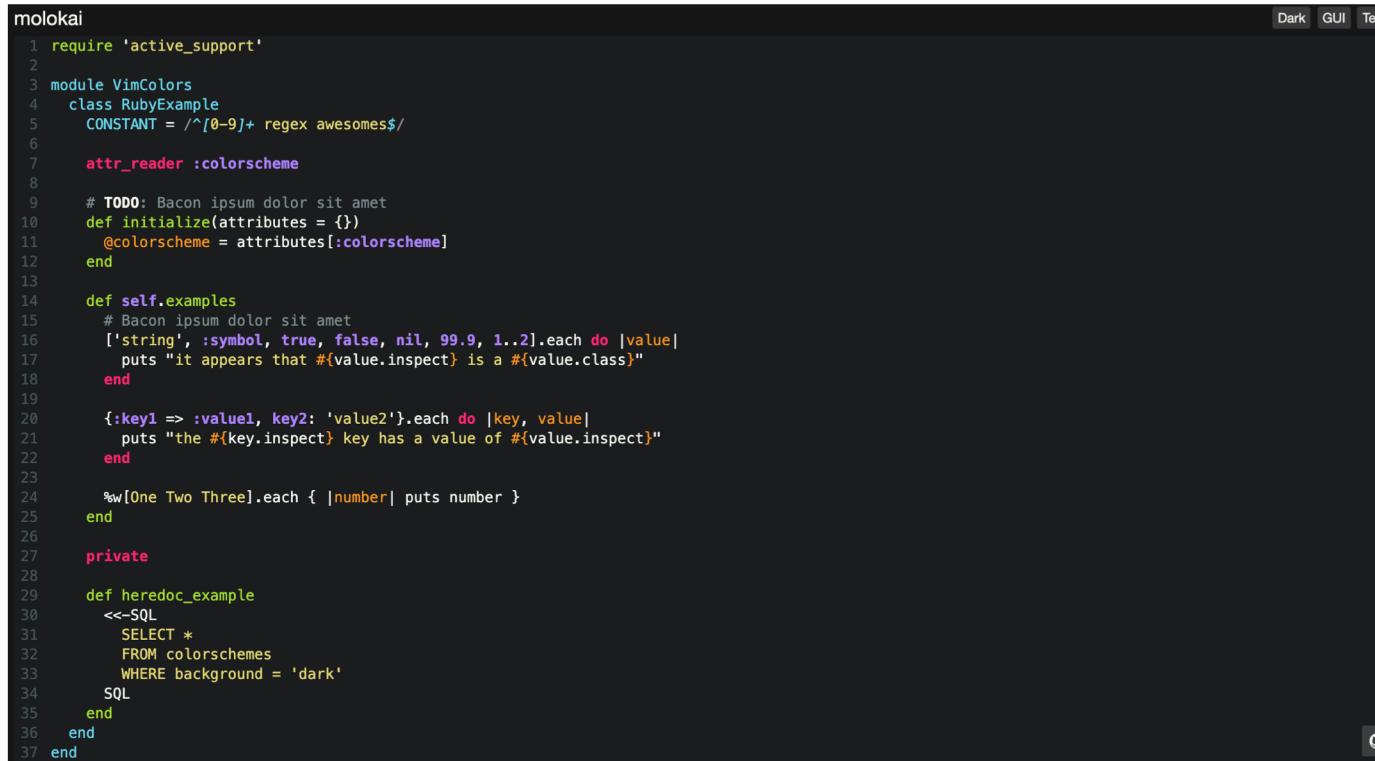
jellybeans

Base 16

など

インポート  
or  
ダウンロード  
して使用

# 試しにmolokaiを使用してみる



A screenshot of a terminal window titled "molokai". The window displays a block of Ruby code with syntax highlighting. The code includes require statements, module definitions, class declarations, and various methods like initialize and examples. The background is dark, and the text is colored in shades of green, yellow, and orange to represent different language constructs. At the top right of the terminal window, there are three tabs: "Dark", "GUI", and "Term".

```
molokai
1 require 'active_support'
2
3 module VimColors
4   class RubyExample
5     CONSTANT = /^[0-9]+ regex awesomes$/
6
7   attr_reader :colorscheme
8
9   # TODO: Bacon ipsum dolor sit amet
10 def initialize(attributes = {})
11   @colorscheme = attributes[:colorscheme]
12 end
13
14 def self.examples
15   # Bacon ipsum dolor sit amet
16   ['string', :symbol, true, false, nil, 99.9, 1..2].each do |value|
17     puts "it appears that #{value.inspect} is a #{value.class}"
18   end
19
20   { :key1 => :value1, key2: 'value2' }.each do |key, value|
21     puts "the #{key.inspect} key has a value of #{value.inspect}"
22   end
23
24   %w[One Two Three].each { |number| puts number }
25 end
26
27 private
28
29 def heredoc_example
30   <<-SQL
31   SELECT *
32   FROM colorschemes
33   WHERE background = 'dark'
34   SQL
35 end
36 end
37 end
```

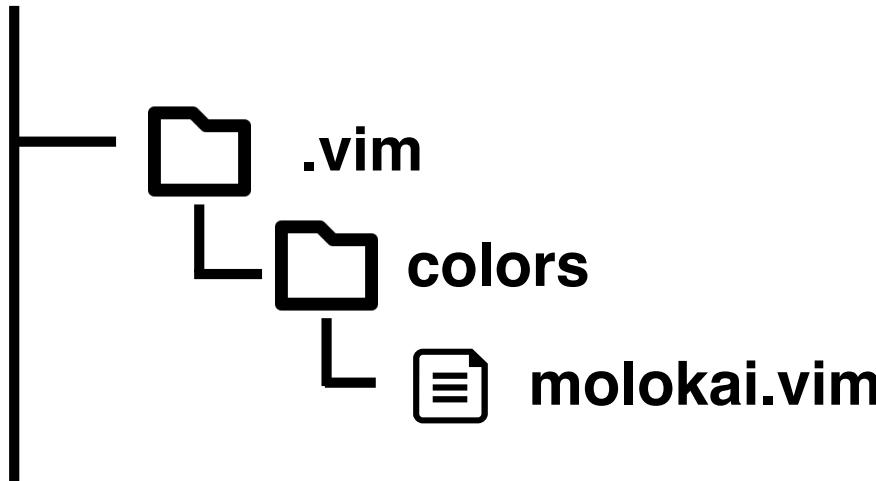
<https://github.com/tomasr/molokai>

# 試しにmolokaiを使ってみる

ダウンロードして使用

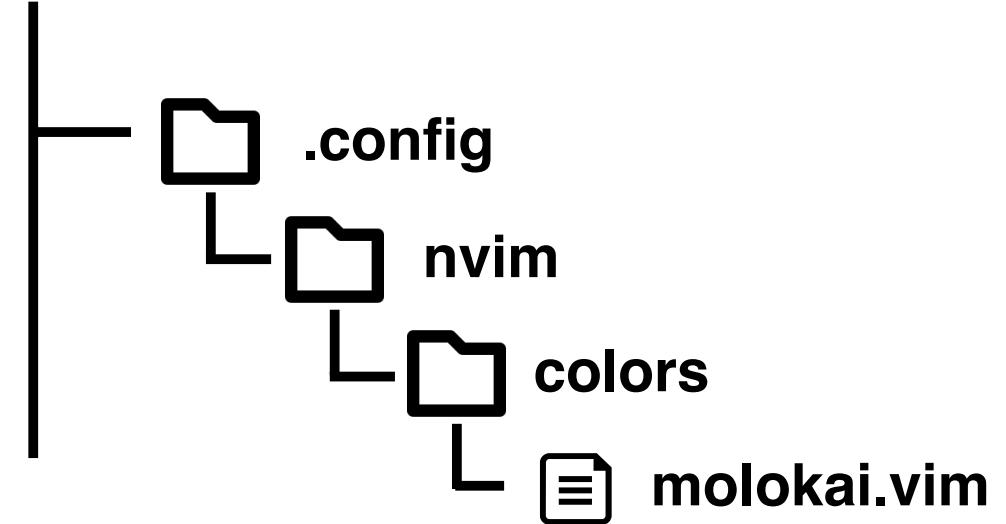
vim

\$HOME



neovim

\$HOME

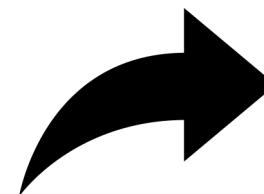


# 試しにmolokaiを使ってみる

ダウンロードして使用

```
syntax enable
```

```
colorscheme molokai
```



```
22
23 " accelerated-jk
24 nmap j <Plug>(accelerated_jk_gj)
25 nmap k <Plug>(accelerated_jk_gk)
26
27 " enable plugin, indent
28 filetype plugin indent on
29 " }}}
30
31 " Basic settings {{{
32 source ~/dev/dotfiles/vim_modules/.vimrc_basic_settings
33
34 " gui configuration
35 highlight Visual term=reverse cterm=reverse guibg=Grey
36 highlight IndentGuidesOdd ctermbg=240
37 highlight IndentGuidesEven ctermbg=238
38
39
40 " Show highlight group under cursor
41
42 source ~/dev/dotfiles/vim_modules/.vimrc_highlight_groups
43
44 " }}}
45
46 " Check whether plugins should be installed or not
47 if has('vim_starting') && dein#check_install()
48 call dein#install()
49 endif
```

だけど…

新しいカラースキームを設定するたびに,  
これを行うのは時間がかかるし、面倒臭い



# プラグインを用いて設定可能！

deinでmolokaiを導入した例

```
call dein#add('tomasr/molokai')
syntax enable
colorscheme molokai
```

これだけで  
大丈夫！

ディレクトリを作らずに、gitのレポジトリから**自動的に取得**できる！



# レベル3

# 公開されたカラースキームの編集



# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. 自作カラースキームを公開する

お気に入りがあるが…

行番号の色が気に入らない  
好きな言語に対応していない  
もう少し、赤をビビッドにしたい

などなど



## 豆知識 ②

### ターミナルの種類について

- ◎ **term**: 白と黒に対応したターミナル
  - ◎ **cterm**: 256色に対応したターミナル
  - ◎ **GUI (Graphical User Interface )**:  
様々な色に対応したターミナル
- 一般的にはこのどちらか



背景に色をつける場合：  
**ctermbg, guibg**

文字に色をつける場合：  
**ctermfg, guifg**

fg: foreground  
bg: background

# 豆知識 ③

- guiは hex で設定可能

- 赤 : #ff0000

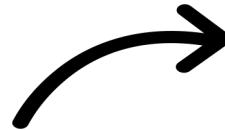
- ctermは xterm256 というパレットで設定

- 赤 : 1

Terminalによっては、ANSIの色の設定を行う必要がある

ある

→ ベースの色が変化する



000000	009900	008700	004600	004798	006000	005000	007700	006000	008700	006000	000000
016	022	028	034	040	045	052	076	070	064	058	052
000002	000502	000752	000502	000752	000502	000502	000502	000502	000502	000502	000502
017	023	029	035	041	047	053	077	071	065	059	053
000007	005807	008707	004607	004797	006007	005007	007707	006007	008707	006007	000007
018	024	030	036	042	048	054	078	072	066	060	054
000011	005401	008741	004641	004794	006041	005041	007741	006041	008741	006041	000011
019	025	031	037	043	049	055	079	073	067	061	055
000047	005847	008747	004647	004797	006047	005047	007747	006047	008747	006047	000047
020	026	032	038	044	050	056	080	074	068	062	056
000008	001508	008708	004608	004798	006008	005008	007708	006008	008708	006008	000008
021	027	033	039	045	051	057	081	075	069	063	057
070000	071500	070700	074600	074798	075000	074500	074798	075000	071500	070000	000000
093	099	106	111	117	123	159	153	147	141	135	129
070007	075807	078707	074607	074797	075007	074507	074797	075007	075807	078707	070007
092	098	104	110	116	122	158	152	146	140	134	128
070046	075506	078746	074606	074796	075006	074506	074796	075006	075506	078746	070046
091	097	103	109	115	121	157	151	145	139	133	127
070087	075807	078707	074607	074797	075007	074507	074797	075007	075807	078707	070087
090	096	102	108	114	120	156	150	144	138	132	126
070052	075502	078752	074652	0747952	075052	074552	0747952	075052	075502	078752	070052
099	095	101	107	113	119	155	149	143	137	131	125
070000	075500	078700	074600	0747900	075000	074500	0747900	075000	075500	078700	070000
098	094	100	106	112	118	154	148	142	136	130	124
d70000	d75000	d77000	d80000	d80000	d80000	d80000	d80000	d80000	d70000	d50000	000000
160	165	172	178	184	190	226	220	214	208	202	196
d70050	d75505	d77505	d80505	d80505	d80505	d80505	d80505	d80505	d70050	d50505	000050
161	167	173	179	185	191	227	221	215	209	203	197
d70087	d75807	d77807	d80807	d80807	d80807	d80807	d80807	d80807	d70087	d50807	000087
162	168	174	180	186	192	228	222	216	210	204	198
d70047	d75507	d77507	d80507	d80507	d80507	d80507	d80507	d80507	d70047	d50507	000047
163	169	175	181	187	193	229	223	217	211	205	199
d70047	d75507	d77507	d80507	d80507	d80507	d80507	d80507	d80507	d70047	d50507	000047
164	170	176	182	188	194	230	224	218	212	206	200
165	171	177	183	189	195	231	225	219	213	207	201
080008	121212	121212	262626	203030	293939	444444	444444	585858	626262	666662	767676
232	233	234	236	236	237	238	239	240	241	242	243
eeeeee	e4e4e4	ddadad	dd00d0	d6d6d6	b6b6b6	b2b2b2	a6a6a6	999999	949494	989898	808080
205	254	253	252	251	250	249	248	247	246	245	244
000000	800000	000000	800000	000000	800000	000000	800000	000000	800000	000000	000000
000	001	002	003	004	005	006	007	008	009	00A	00B
808080	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000	000000
009	009	009	009	011	011	013	014	015			

# 補足

xterm256の色をターミナルで見たい！

<https://github.com/guns/xterm-color-table.vim>

をvimrcにプラグインとして記述すると

**:XtermColorTable**

というコマンドが生成される。これで、みることができる

# どうやって色をつければいいんだ？

色は「ハイライトグループ」ごとに付けられている

**Comment:** コメント

**Constant:** 定数

**String:** 文字列 ("これは文字列です")

**Character:** 文字定数 ('c', '\n')

**Number:** 数値定数 (234, 0xff)

**Statement:** 命令文

**Conditional:** if, then, else, endif...

などなど

# だからと言って、わからん！

## お助けコマンド ①

vim上で

```
:highlight
```

もしくは

```
:hi
```

とコマンドを打つと、

グループ名 表示内容 色

のように現在の状態を出力する

```
SpecialKey ctermfg=22 ctermbg=236 guifg=#f1530f guibg=#1a1a1a
EndofBuffer links to NonText
TermCursor cterm=reverse gui=reverse
TermCursorNC xxx cleared
NonText ctermfg=22 ctermbg=234 guifg=#30312a guibg=#1a1a1a
Directory xxx ctermfg=15 ctermbg=9 guifg=#fffffff guibg=#ff0000
ErrorMsg xxx ctermfg=15 ctermbg=9 guifg=#6c71c4
IncSearch cterm=bold ctermfg=16 ctermbg=39 gui=bold guifg=#000000 guibg=#6c71c4
Search cterm=bold ctermfg=15 ctermbg=196 gui=bold guifg=#f7f3ff guibg=#e1d4d2
MoreMsg xxx ctermfg=121 gui=bold guifg=SeaGreen
ModeMsg xxx cterm=bold gui=bold
LineNr xxx ctermfg=242 ctermbg=234 guifg=#838586 guibg=#2f2f2f
CursorLineNr xxx ctermfg=11 gui=bold guifg=Yellow
Question xxx ctermfg=121 gui=bold guifg=Green
StatusLine cterm=bold ctermfg=231 ctermbg=240 gui=bold guifg=#ecf0f1 guibg=#575858
StatusLineNC xxx ctermfg=231 ctermbg=240 guifg=#ecf0f1 guibg=#575858
VertSplit ctermfg=244 ctermbg=234 guifg=#575858 guibg=#575858
Title cterm=bold ctermfg=231 gui=bold guifg=#ecf0f1
Visual cterm=reverse ctermbg=238 guibg=Grey
VisualNC xxx cleared
WarningMsg xxx ctermfg=15 ctermbg=9 guifg=#fffffff guibg=#ff0000
WildMenu xxx ctermfg=8 ctermbg=11 guifg=Black guibg=Yellow
Folded ctermfg=241 ctermbg=234 guifg=#606060 guibg=#1a1a1a
FoldColumn xxx ctermfg=14 ctermbg=242 guifg=Cyan guibg=Grey
DiffAdd cterm=bold ctermfg=231 ctermbg=64 gui=bold guifg=#ecf0f1 guibg=#44800a
DiffChange xxx ctermfg=231 ctermbg=23 guifg=#ecf0f1 guibg=#1d3251
DiffDelete ctermfg=88 guifg=#88005
DiffText xxx cterm=bold ctermfg=231 ctermbg=40 gui=bold guifg=#ecf0f1 guibg=#00df00
SignColumn xxx ctermfg=244 ctermbg=236 guifg=#838586 guibg=#2f2f2f
links to LineNr
Conceal ctermfg=7 ctermbg=242 guifg=LightGrey guibg=DarkGrey
SpellBad xxx ctermfg=9 ctermbg=224 gui=undercurl guisp=Red
SpellCap xxx ctermbg=12 gui=undercurl guisp=Blue
SpellRare xxx ctermbg=13 gui=undercurl guisp=Magenta
SpellLocal xxx ctermbg=14 gui=undercurl guisp=Cyan
Pmenu xxx ctermfg=41 guifg:#2ecc71
PmenuSel xxx ctermbg=238 guibg=#444444
PmenuSbar xxx ctermbg=248 guibg=Grey
PmenuThumb xxx ctermbg=15 guibg=White
Tabline cterm=underline ctermfg=15 ctermbg=242 gui=underline guibg=DarkGrey
TablineSel xxx cterm=bold gui=bold
TablineFill xxx cterm=reverse gui=reverse
CursorColumn xxx ctermbg=236 guifg=#2f2f2f
CursorLine xxx ctermbg=236 guifg=#2f2f2f
ColorColumn xxx ctermbg=236 guibg=#2f2f2f
QuickFixLine xxx links to Search
Whitespace xxx links to NonText
NormalNC xxx cleared
MsgSeparator xxx links to StatusLine
Cursor ctermfg=234 ctermbg=231 guifg=#1a1a1a guibg=#ecf0f1
lCursor xxx guifg=bg guibg=fg
Substitute xxx links to Search
MatchParen xxx cterm=underline ctermfg=9 gui=underline guifg=#ff0000
Normal xxx ctermfg=231 ctermbg=234 guifg=#ecf0f1 guibg=#1a1a1a
NvimInternalError xxx ctermfg=9 ctermbg=9 guifg=Red guibg=Red
NvimAssignment xxx links to Operator
Operator xxx cterm=bold ctermfg=9 gui=bold guifg=#ff0000
NvimPlainAssignment xxx links to NvimAssignment
NvimAugmentedAssignment xxx links to NvimAssignment
NvimAssignmentWithAddition xxx links to NvimAugmentedAssignment
NvimAssignmentWithSubtraction xxx links to NvimAugmentedAssignment
-- 続続 --
```

# それでもよくわからん！

## お助けコマンド ②

<https://github.com/sff1019/dotfiles/> の  
vim\_modules/.vimrc\_syntax\_info を .vimrc 内に記述すると

:SyntaxInfo

というコマンドが使用できるようになる

```
name: vimCommand ctermfg: ctermbg: guifg: guibg:  
link to  
name: Statement ctermfg: 9 ctermbg: guifg: #ff0000 guibg:  
続けるにはENTERを押すかコマンドを入力してください■
```

# 試しにいじってみる

vim上で色をいじる際のコマンド

:highlight もしくは :hi

vim上で

```
:hi Normal ctermfg=1 guibg=#ff0000
```

というコマンドを実行する



```
" vimrc
" Only for nvim
if !has('nvim')
  set ttymouse=xterm2
endif
" Flags {{
let $use_dein = 1
" }}

" Filetype Settings
source ~/dev/dotfiles/vim_modules/ vimrc_filetype

" Dein settings
source ~/dev/dotfiles/vim_modules/ vimrc_dein

" Plugin settings {{
" accelerated-jk
nnmap j <Plug>(accelerated_jk_gj)
nnmap k <Plug>(accelerated_jk_gk)

" enable plugin, indent
filetype plugin indent on
" }}

" Basic settings {{
```



# 全体に反映したい！

- ① .vimrcに直接変更内容を書く
- ② カラースキームファイルを編集

# 実際にいじってみる

~.vimrcに直接内容を書く~

1. colorschemeの前に、変更内容を書き加える
2. 例えば、行番号の色を変えたい時：

```
autocmd ColorScheme * highlight LineNr\
ctermbg=15 ctermfg=1 guibg=#ffffff guifg=#ff0000
```

とし、vimを再起動すると行番号の背景と文字の色が変わる

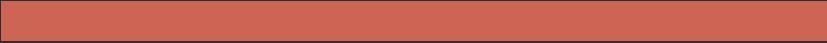
# 実際にいじってみる

## ～カラースキームファイルを編集～

1. カラースキームファイルを取得
2. 下記のように、変更を加える

```
hi LineNr ctermbg=15 ctermfg=1\
guibg=#ffffffff guifg=#ff0000
```

3. Github\*に追加、もしくは /colorsディレクトリに追加



# レベル4 自作カラースキーム



# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. 自作カラースキームを公開する

求めているカラースキームがない…

悩むくらいなら作ってしまおう！



# カラースキームを作成する手順

- ① テーマの設定
- ② メインの色を決める
- ③ 編集を行う
- ④ 実際のコードに反映し、手直し
- ⑤ Github等に公開・colors/内に置く

## 豆知識 ④

実際、1から手作業でカラースキームを作るのは大変

<http://bytefluent.com/vivify/index.php>

という、カラースキームオンラインエディタが  
非常に便利！

## 豆知識 ⑤

様々な言語で、試した方がより綺麗な

カラースキームが作成できる

<https://github.com/acmeism/RosettaCodeData>

のレポジトリには様々な言語が入っているので、

試す際に非常に役立つ！

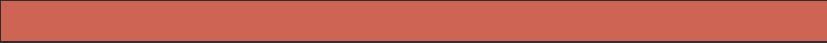
## 豆知識 ⑥

色合いなど、詳しくない人は

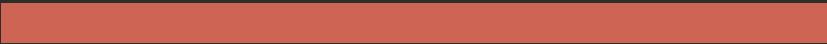
<https://coolors.co/>

のような、オンラインカラースキームジェネレタを  
用いると、簡単に色の配列が得られる。

また、画像からも抽出できる。



# レベル5 自作カラースキームを公開



# カラースキーマレベル

1. デフォルトのカラースキームを使用
2. 公開されたカラースキームを使用
3. 公開されたカラースキームをカスタマイズして使用
4. 自作カラースキームを使用
5. **自作カラースキームを公開する**

# あとは公開するだけ！

しかし、Githubに公開する際に、  
**ディレクトリ構造**に気をつけるべき！

プラグインツールは**決まったディレクトリ構造**から、  
カラースキームを読み取る

# ディレクトリ構造

- ・階層に気をつける
- ・colorsというディレクトリ内に  
vimファイルを作成



# おすすめサイト

<http://colorswat.ch/vim/>

- ・**厳選された**カラースキームがある
- ・サイト作成者自身、有名なカラースキーム、Icebergを作成

<https://cocopon.me/blog/2016/02/iceberg/>

- 先ほどのサイト作成者がカラースキームを作成した時の考慮した点などを丁寧に紹介している

# My Colorscheme

- <https://github.com/sff1019/vim-brogrammer-theme>



A screenshot of the Vim text editor displaying a Python script named `example.py`. The code implements a command-line interface to build a vimrc file. The Vim interface includes a status bar at the bottom and a buffer list titled "buffers" at the top right. The code uses a dark color scheme with syntax highlighting for keywords (e.g., `import`, `def`) and punctuation.

```
vimrc.vim > example.py+ buffers
1 import sys
2 import os
3 import os.path
4 from pathlib import Path
5 from vint.linting.file_filter import find_vim_script
6
7
8 def build_environment(cmdargs):
9     return {
10         'cmdargs': cmdargs,
11         'home_path': _get_home_path(),
12         'xdg_config_home': _get_xdg_config_home(),
13         'cwd': _get_cwd(),
14         'file_paths': _get_file_paths(cmdargs),
15     }
16
17
18 def _get_cwd():
19     return Path(os.getcwd())
20
21
22 def _get_home_path():
23     return Path(os.path.expanduser('~/'))
24
25
26 def _get_file_paths(cmdargs):
27     if 'files' not in cmdargs:
28         return []
29
30     found_file_paths = find_vim_script(map(Path, cmdargs['files']))
31
32     return found_file_paths
33
34
```

# My Colorscheme

- <https://github.com/sff1019/vim-joker>



<https://nerdfonts.com/>

- カラースキーム以外にも、フォント等を変えたい人向け
- Status Lineの絵文字や、ファイルの絵文字を変える際に非常に便利

# Reference

- Molokai  
<https://github.com/tomasr/molokai>
- 構文ハイライト  
<https://vim-jp.org/vimdoc-ja/syntax.html>
- カラースキーム作成  
<http://bytefluent.com/vivify/>
- Xterm256  
<https://github.com/guns/xterm-color-table.vim>
- イラスト  
<https://www.irasutoya.com/>
- Vimのドキュメント  
<https://www.vim.org/docs.php>



ありがとうございます

