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**

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…

```
learning_rate = 0.001
batch_size = 50
step_size = 3

# Fully connected neural network with one hidden layer


class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size * 2)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size * 2, hidden_size)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu1(out)
        out = self.fc2(out)
        out = self.relu2(out)
        out = self.fc3(out)
        return out


model = NeuralNet(input_size, hidden_size).to(device)

# Loss and optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

train_dataset = list(range(0, step_size))
# val_dataset = list(range(0, 26))

print('start training')
# Train the model
for epoch in range(num_epochs):
    for i in train_dataset:
        # Move tensors to the configured device
        filename = 'models/preprocessed_{}.pickle'.format(i)
        with open(filename, 'rb') as f:
            print('loading data')
            train_data = pickle.load(f).astype(np.float32)
            print('finish_loading')

```

見えずらい・何が何だかわかりずらい・かっこ悪い

カラースキームを追加すると…

```
learning_rate = 0.001
batch_size = 50
step_size = 3

# Fully connected neural network with one hidden layer


class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(NeuralNet, self).__init__()
        self.fc1 = nn.Linear(input_size, hidden_size * 2)
        self.relu1 = nn.ReLU()
        self.fc2 = nn.Linear(hidden_size * 2, hidden_size)
        self.relu2 = nn.ReLU()
        self.fc3 = nn.Linear(hidden_size, 1)

    def forward(self, x):
        out = self.fc1(x)
        out = self.relu1(out)
        out = self.fc2(out)
        out = self.relu2(out)
        out = self.fc3(out)
        return out


model = NeuralNet(input_size, hidden_size).to(device)

# Loss and optimizer
criterion = nn.MSELoss()
optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate)

train_dataset = list(range(0, step_size))
# val_dataset = list(range(0, 26))

print('start training')
# Train the model
for epoch in range(num_epochs):
    for i in train_dataset:
        # Move tensors to the configured device
        filename = 'models/preprocessed_{}.pickle'.format(i)
        with open(filename, 'rb') as f:
            print('loading data')
            train_data = pickle.load(f).astype(np.float32)
            print('finish_loading')

```

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  word
```

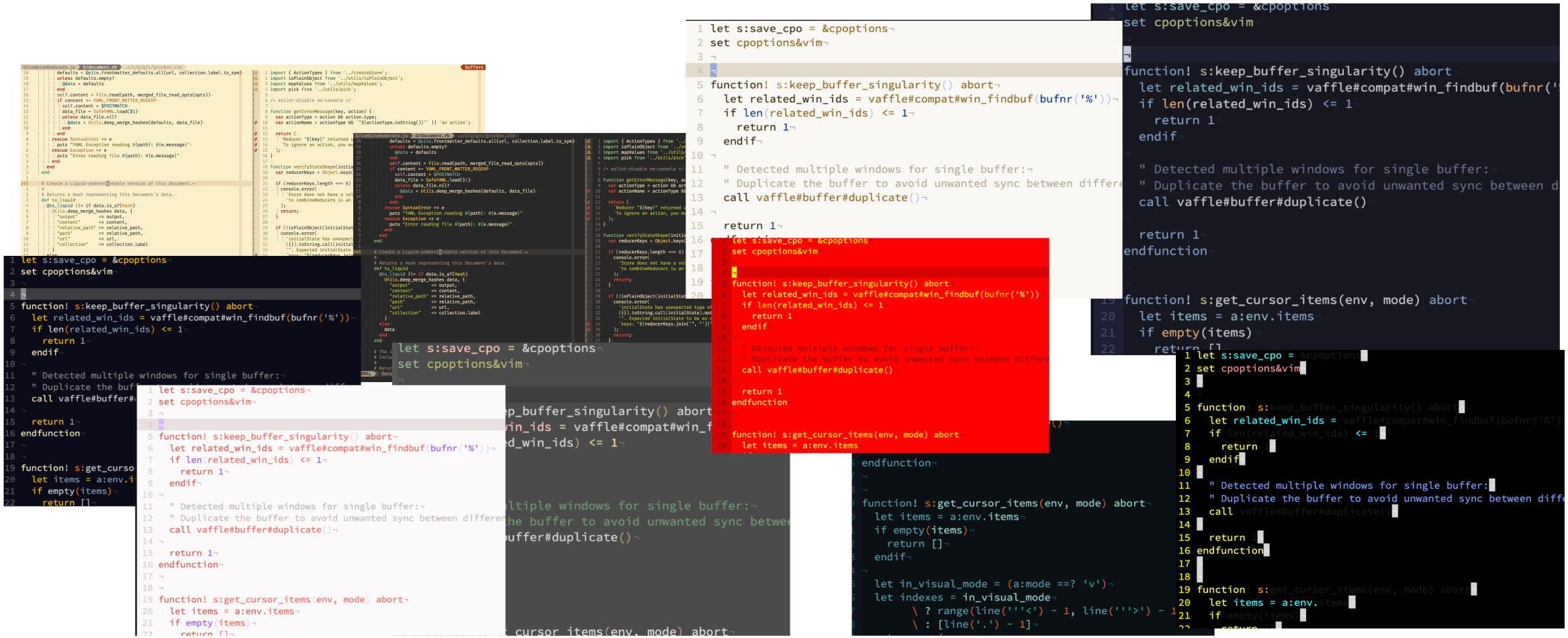
```python
import csv
import sys
import re
import time
import pickle
from itertools import chain

import numpy as np


if __name__ == '__main__':

    path = 'mercari/test_stg2.tsv'
    # path = 'mercari/train.tsv'
    f = open(path, 'r', encoding='utf-8')
    tsv = csv.reader(f, delimiter='\t')
    next(tsv)

    with open('processed/ohe_item_condition
        ohe_condition = pickle.load(f)

    with open('processed/le_item_condition_
        le_condition = pickle.load(f)

    with open('processed/word2vec_name.pick
        word2vec_name = pickle.load(f)

    with open('processed/word2vec_item_desc
        word2vec_item_description = pickle.

    with open('processed/word2vec_brand_nam
        word2vec_brand_name = pickle.load(f)

    with open('processed/word2vec_category_name.pickle', 'rb') as f:
        word2vec_category_name = pickle.load(f)

    print('Reading...')
    sys.stdout.flush()
    rows = []
    start_time = time.time()
    count = 0
    for index, line in enumerate(tsv):
        condition = le_condition.transform([line[2]]).reshape(1, 1)
        np_condition = ohe_condition.transform(condition).flatten()

        name = [s for s in [s.strip().lower().replace('/', ' ').split(' ')
                            for s in re.split('\.', line[1])]]
        name = list(chain.from_iterable(name))
        np_name = word2vec_name.wv[name].mean(axis=0).flatten()

        # desc = [s for s in [s.strip().lower().replace('/', ' ').split(' ')
```

```
vim 10.0 %
format.py  format_individual.py  eval.py  preprocess.py  submit.py  word2
```

```python
import csv
import sys
import re
import time
import pickle
from itertools import chain

import numpy as np


if __name__ == '__main__':

    path = 'mercari/test_stg2.tsv'

    with open('processed/word2vec_category_name.pickle', 'rb') as f:
        word2vec_category_name = pickle.load(f)

    print('Reading...')
    sys.stdout.flush()
    rows = []
    start_time = time.time()
    count = 0
    for index, line in enumerate(tsv):
        condition = le_condition.transform([line[2]]).reshape(1,
        np_condition = ohe_condition.transform(condition).flatte

        name = [s for s in [s.strip().lower().replace('/', ' ').
                            for s in re.split('\.', line[1])]]
        name = list(chain.from_iterable(name))
        np_name = word2vec_name.wv[name].mean(axis=0).flatten()

        # desc = [s for s in [s.strip().lower().replace('/', ' ').s
        #                     for s in re.split('\.', line[7])]]
```

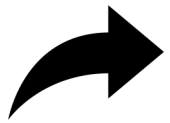
カラースキームは偉大！

もちろん種類はたくさんあるのが vimの良いところ！



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



著作権等に注意

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]`

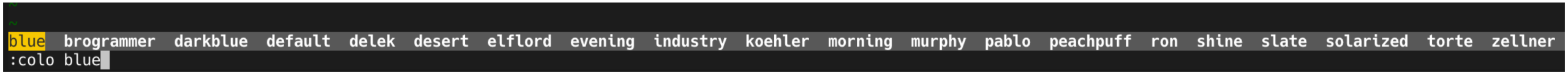
豆知識 ①

vim上で現在あるカラースキームの種類がわかる！

`:coloscheme` SPACE TAB

または

`:colo` SPACE TAB



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
```

再起動するとこうなるはず！

```
" Fastest way to move buffer
nnoremap <silent><Left> :bp<CR>
nnoremap <silent><Right> :bn<CR>
nnoremap <silent><C-Space> :call BufferDeleteExceptFiler()<CR>

function! BufferDeleteExceptFiler()
  if (&filetype !=# 'vimfiler')
    bd!
  endif
endfunction

if !has('gui_running')
  augroup term_vim_c_space
    autocmd!
    autocmd VimEnter * map <Nul> <C-Space>
    autocmd VimEnter * map! <Nul> <C-Space>
  augroup END
endif

" Display another buffer when current buffer isn't saved.
set hidden

" Do not create swap files
set noswapfile

" Set clipboard
set clipboard+=unnamedplus


" 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　　hybrid

gruvbox　　jelleybeans

solarized　　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

molokai Dark GUI Term

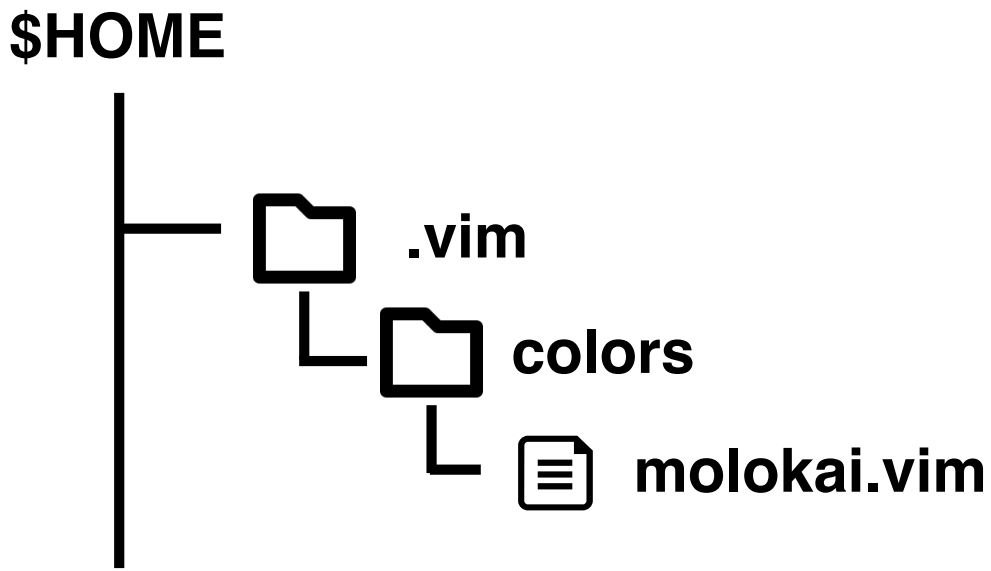
```ruby
require 'active_support'

module VimColors
  class RubyExample
    CONSTANT = /^[0-9]+ regex awesomes$/

    attr_reader :colorscheme

    # TODO: Bacon ipsum dolor sit amet
    def initialize(attributes = {})
      @colorscheme = attributes[:colorscheme]
    end

    def self.examples
      # Bacon ipsum dolor sit amet
      ['string', :symbol, true, false, nil, 99.9, 1..2].each do |value|
        puts "it appears that #{value.inspect} is a #{value.class}"
      end

      {:key1 => :value1, key2: 'value2'}.each do |key, value|
        puts "the #{key.inspect} key has a value of #{value.inspect}"
      end

      %w[One Two Three].each { |number| puts number }
    end

    private

    def heredoc_example
      <<-SQL
        SELECT *
        FROM colorschemes
        WHERE background = 'dark'
      SQL
    end
  end
end
```

https://github.com/tomasr/molokai

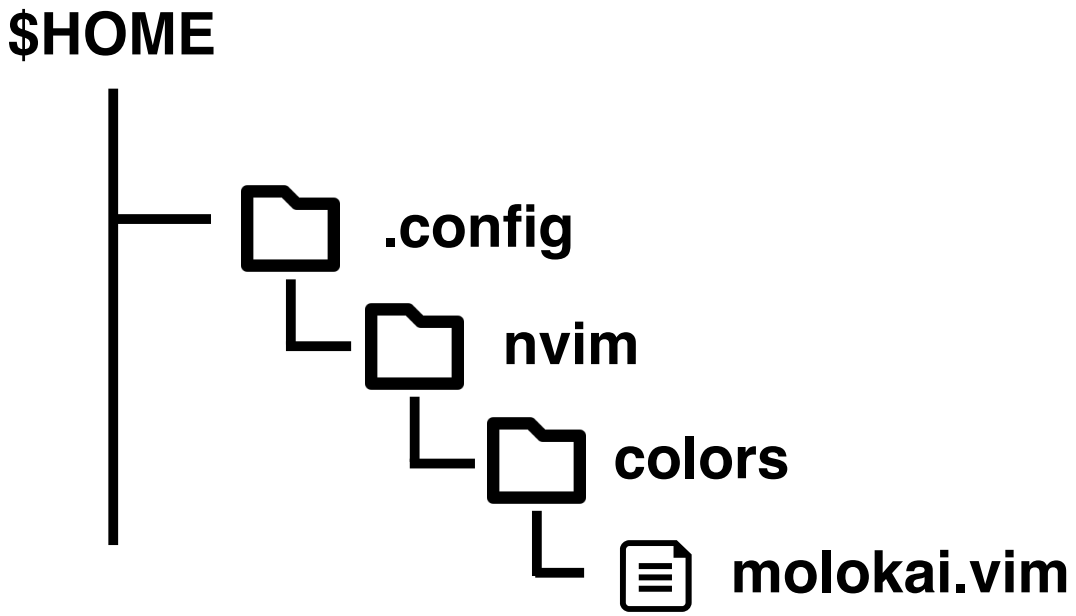
試しにmolokaiを使ってみる

ダウンロードして使用

vim



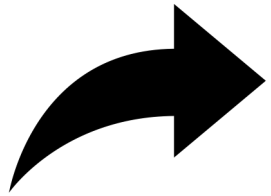
neovim



試しにmolokaiを使ってみる

ダウンロードして使用

```
syntax enable

colorscheme molokai
```



```
" accelerated-jk
nmap j <Plug>(accelerated_jk_gj)
nmap k <Plug>(accelerated_jk_gk)

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

" Basic settings {{{
source ~/dev/dotfiles/vim_modules/.vimrc_basic_settings

" gui configuration
highlight Visual term=reverse cterm=reverse guibg=Grey
highlight IndentGuidesOdd  ctermbg=240
highlight IndentGuidesEven ctermbg=238

"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
" Show highlight group under cursor
"""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""""
source ~/dev/dotfiles/vim_modules/.vimrc_highlight_groups

" }}}

" Check whether plugins should be installed or not
if has('vim_starting') && dein#check_install()
  call dein#install()
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

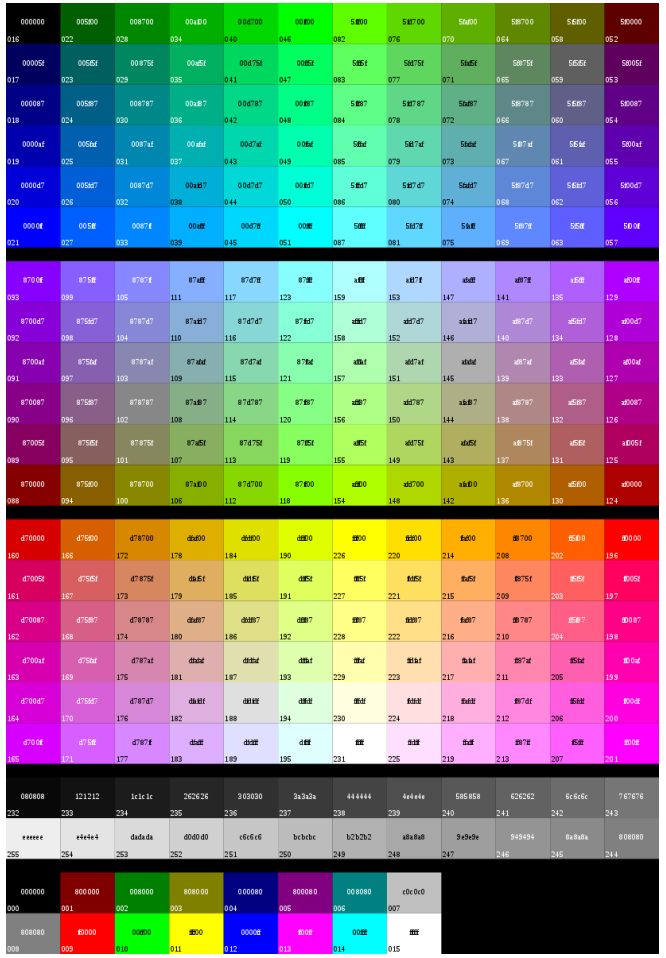
背景に色をつける場合：
`ctermbg, guibg`

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

→ ベースの色が変化する

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

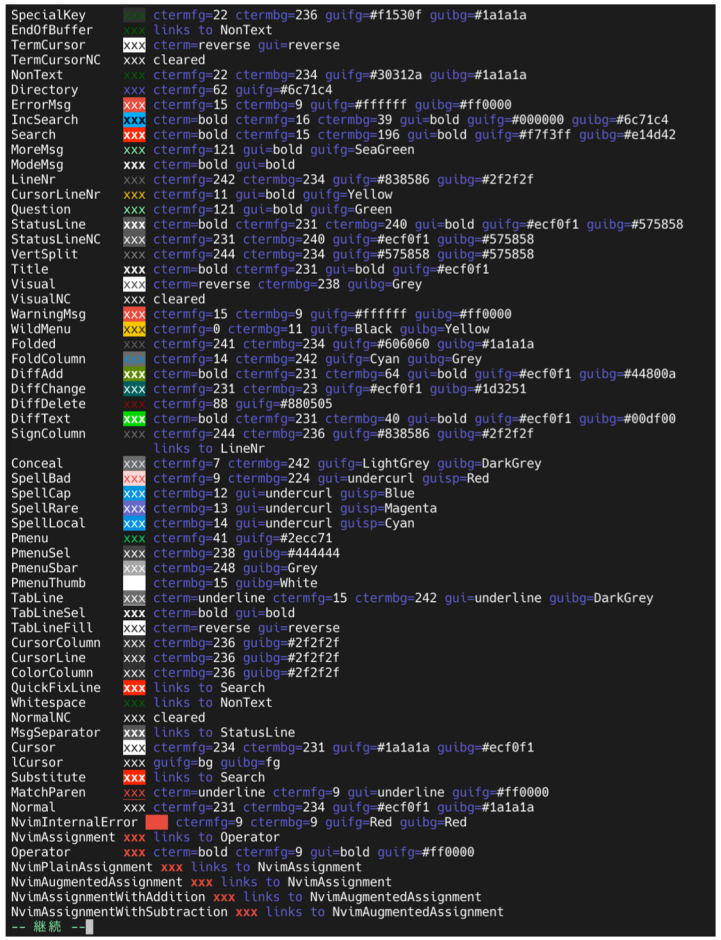
vim上で

`: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



それでもよくわからん！

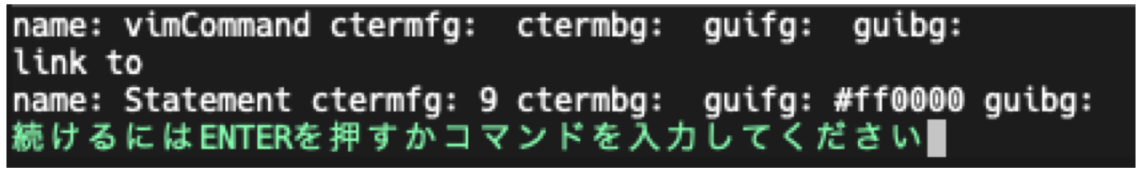
お助けコマンド ②

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



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 ctermbg=1 guibg=#ff0000
```

というコマンドを実行する

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=#ffffff 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ータを

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

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，ファイラー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