**『独学プログラマー』メモ**

無限ループを止めるには

Ctrl－c

または

【Fn】－【Break】

現在のフォルダーを見るには

import os

print(os.getcwd())

sudoコマンドはルートユーザー権限で実行すること！

GitBashのコピー、ペーストは

「Ctrl+insert」コピー

「shift+insert」ペースト

【プログラミングパラダイム】

1. 手続き型プログラミング

状態をすべてグローバル変数にもたせている。プログラムが大きくなり、予期しないミスが生じやすくなる。コードがグローバル変数の状態を変えてしまうという副作用がある。

1. 関数型プログラミング

副作用なし。関数がグローバル状態に依存せず、関数に渡された引数のみによって動作が変わる。グローバル状態を持たせた方が解決しやすい問題は扱いづらい。

1. オブジェクト指向プログラミング

副作用なし。オブジェクトに状態をもたせる。複数のクラスで相互作用するオブジェクトの集合となる。すべてのオブジェクトがクラスのインスタンスとなる。インスタンスごとに属性が異なる。

【オブジェクト指向プログラミングの4大要素＋１】

1. カプセル化
2. 抽象化
3. ポリモーフィズム
4. 継承
5. コンポ十ション

【BASH（コマンドラインインターフェース）】

キーボードで命令を入力してプログラムを対話形式で実行する。

コマンド：

1. pwd：現在のディレクトリを表示
2. ls：現在のディレクトリ内のディレクトリ名やファイル名（ls –aで隠しファイル表示）
3. cd：ディレクトリの変更（cd ..で一つ下に下がる）
4. mkdir：新しいディレクトリの作成
5. rmdir：ディレクトリの削除
6. whoami：ユーザー名の表示
7. grep：文字列の検索
8. pip：パッケージのインストール、アンインストール

ｌｓ－－あうてょｒ－ｌｌｓ

【正規表現】

文字列の検索

まず、次のコマンドで、“The Zen of Python”という詩を表示させる。

C:\Users\yokan\desktop> python -c "import this"

・・・

The Zen of Python, by Tim Peters

Beautiful is better than ugly.

Explicit is better than implicit.

Simple is better than complex.

Complex is better than complicated.

Flat is better than nested.

Sparse is better than dense.

Readability counts.

Special cases aren't special enough to break the rules.

Although practicality beats purity.

Errors should never pass silently.

Unless explicitly silenced.

In the face of ambiguity, refuse the temptation to guess.

There should be one-- and preferably only one --obvious way to do it.

Although that way may not be obvious at first unless you're Dutch.

Now is better than never.

Although never is often better than \*right\* now.

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

以下も追加しておく：

123 hi 34 hello.

2\*3=6

・・・

この詩をデスクトップ上の“zen.txt”の中に収める。

Grepコマンドで一致した文字を赤色で示せるように次の命令を実行しておく。

$ export GREP\_OPTIONS='--color=always'

例えば、zen.txtの中で”Beautiful”を含む行を出力するには、

$ grep Beautiful zen.txt

Beautiful is better than ugly.

大文字と小文字の違いを無視するには、

$ grep –i Beautiful zen.txt

Beautiful is better than ugly.

行の先頭に”If”が来る行を表示するには、”^”をつける。

$ grep ^If zen.txt

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

“idea”の次にどんな文字が来ても表示してくれるためには“.”をつける。

$ grep idea. zen.txt

If the implementation is hard to explain, it's a bad idea.

If the implementation is easy to explain, it may be a good idea.

Namespaces are one honking great idea -- let's do more of those!

数字のある行を表示するには、[[:digit:]]を使う。

$ grep [[:digit:]] zen.txt

123 hi 34 hello.

2\*3=6

【パッケージ管理】

pipによるパッケージ“Flask”最新版のインストール。

$ pip install Flask

また、あるパッケージをアンインストールするには、

$ pip uninstall “パッケージ名”

パッケージの一覧を見るには、

$ pip freeze

ウェブサイト作成モジュール「Flask」をインポートするコード“h59sdyu.py”を実行する。

・・・h59sdyu.pyのコード・・・

from flask import Flask

app = Flask(\_\_name\_\_)

@app.route('/')

def index():

return "Hello, World!"

app.run(port='8000')

・・・

・・・h59sdyu.pyの実行とその結果・・・

C:\Users\yokan\desktop> python h59sdyu.py

\* Running on http://127.0.0.1:8000

Press CTRL+C to quit

・・・

ここで、CTRL+Cを押す前に、ウェブブラウザでURL“<http://127.0.0.1:8000>”を入力すると、

Hello, World!

が表示される。

【バージョン管理】

GitHubのアカウント作成は

[http://windows.github.com](http://windows.github.com/)

からダウンロードして行う。

Username: yokando

Password: ht195428

リポジトリURL: <https://github.com/yokando/hangman>

さらに、次のサイトを参照して、Gitをインストールする。

<https://www.git-scm.com/book/ja/v2/>使い始める-Gitのインストール

次のコマンドを実行すると、ローカルリポジトリ“hangman”が出来る。

$ git clone <https://github.com/yokando/hangman>

次のコマンドで中央リポジトリを確認する。

C:\Users\yokan\hangman> git remote -v

origin https://github.com/yokando/hangman (fetch)

origin https://github.com/yokando/hangman (push)

プッシュ：ローカルリポジトリの内容を中央リポジトリに同期させる

ローカルリポジトリC:\Users\yokan\hangmanにファイル“hangman.py”を置く。

$ git add hangman.py　：ファイルをステージ状態にする

$ git commit –m “my first commit”　：コミットに説明文を付け加える

$ git push origin main　：ステージ状態のファイルを中央リポジトリにプッシュする

（㊟git push origin masterでなくgit push origin mainでないとうまく行かない）

プル：中央リポジトリに記録された新しい差分をローカルリポジトリに同期させる

まず、GitHubサイトの「Add file」から「Create new file」ボタンを押して、new.pyという名前を入力し、「Commit changes」から「Commit changes」ボタンを押すと、中央リポジトリにnew.pyファイルが新たに作成される。そこで、GitBushの画面から

$ git pull origin main　：中央リポジトリの変更をローカルリポジトリに取り込む

（㊟git pull origin masterでなくgit pull origin mainでないとうまく行かない）

ここで、ローカルリポジトリにあるファイルを見てみる。

$ ls

hangman.py new.py README.md

次に、リポジトリの履歴を見てみる。

$ git log

commit 5d77420bfc4f93aa9454d8e34cdd4bdea236c371 (HEAD -> main, origin/main, origin/HEAD)

Author: yokando <157599229+yokando@users.noreply.github.com>

Date: Thu Jan 25 15:52:29 2024 +0900

Create new.py

commit da3dd6e61c5eb6251de516ede7155f74ac5db088

Author: yokando <yokando2@gmail.com>

Date: Thu Jan 25 06:27:25 2024 +0900

my first commit

commit e166653d0824813154db928a10dc0cef4359176a

Author: yokando <157599229+yokando@users.noreply.github.com>

Date: Wed Jan 24 20:51:33 2024 +0900

Initial commit

ここで、“Initial commit”のコミット番号を使って最初の状態に戻す。

$ git checkout e166653d0824813154db928a10dc0cef4359176a

再び、ローカルリポジトリにあるファイルを見てみる。

$ ls

README.md

これで、最初の状態となった。しかし、中央リポジトリはそのまま3ファイル。

次に、ファイルhello.pyを作成し、そこに“Hello,”と書いて、ローカルリポジトリに置く。そして、ステージ状態にし、コミットする。hello.pyを“hello, World!”に書き換える。そこで、diffコマンドで差分を見る。以下の通りなり。

$ git add hello.py

$ git commit –m “adding new file”

$ git diff hello.py

diff --git a/hello.py b/hello.py

index 0f852de..273b5b5 100644

--- a/hello.py

+++ b/hello.py

@@ -1 +1 @@

-Hello,

\ No newline at end of file

+Hello, World!

\ No newline at end of file

【ＨＴＭＬ】

次のhtmlコードをファイルに保存して、ダブルクリックすると、ウェブブラウザに“Hello, World!”が表示される。

・・・

<!--This is a comment in HTML. Save this file as index.html-->

<html lang="en">

<head>

<meta charset="UTF-8">

<title>My Website</title>

</head>

<body>

Hello, World!

<a href="https://news.ycombinator.com/">here</a>

</body>

</html>

・・・

次にGoogleニュースをスクレイピングする。

Beautifulsoupのインストール

C:\Users\yokan\hangman> pip install beautifulsoup4==4.6.0

Pythonコード“j55s7hm.py”

・・・

# Thanks so much for reading my book. Feel free to contact me at cory[at]theselftaughtprogrammer.io.

### GOOGLE CHANGED THEIR CODE. Try using 'articles' instead of 'html'

import urllib.request

from bs4 import BeautifulSoup

class Scraper:

def \_\_init\_\_(self, site):

self.site = site

def scrape(self):

r = urllib.request\

.urlopen(self.site)

html = r.read()

parser = "html.parser"

sp = BeautifulSoup(html, parser)

for tag in sp.find\_all("a"):

url = tag.get("href")

if url is None:

continue

if "html" in url:

print("\n" + url)

news = "https://news.google.com/"

Scraper(news).scrape()

・・・

ウェブスクレイパーのコードj55s7hm.pyを実行したところ、エラーが出た。

(base) PS C:\Users\yokan\hangman> python j55s7hm.py

AttributeError: module 'collections' has no attribute 'Callable'

それで、Mypm3の環境で行うことにしたら、うまく行ったが、何も表示されなかった。

(base) PS C:\Users\yokan\hangman> conda activate mypm3env

(mypm3env) PS C:\Users\yokan\hangman> python j55s7hm.py

【データ構造】

スタックの5つのメソッド

is\_empty, push, pop, peek, size

キューの４つのメソッド

enqueuer, dequeuer, is\_empty, size

【アルゴリズム】

線形探索

回文

アナグラム

回帰

【さらに学ぼう】

古典で学ぶ

・アンドリュー・ハントとデイブ・トーマス『達人プログラマー』

・エリック・ガンマー他『オブジェクト指向における再利用のためのデザインパターン』

・スティーブ・マコネル『Code Complete完全なプログラミングを目指して』

・アルフレッド・エイホ他『コンパイラー原理・技法・ツール』

・MITプレス『アルゴリズムイントロダクション』

・ブラッドリー・N・ミラーとデイビッド・L・ラナム『Problem Solving with Data Structure and Algorithms』

オンライン授業で学ぶ

・https://theselftaughtprogrammer.io/courses

・https://www.udemy.com/python-j/　（ノンプログラマーが始めるPython）

・https://www.udemy.com/python-jp/　（実践Pythonデータサイエンス）

ハッカーニュース

・https://news.ycombinator.com

・http://postd.cc/　（ハッカーニュース記事の一部日本語訳）

プログラミングについてのQ&Aサイト

・https://ja.stackoverflow.com/

・https://teratail.com/

日本のPythonコミュニティの一覧

・https://www.python.jp/pages/community.html

プログラミング学習サービスなど

・https://pyq.jp/

・https://projecteuler.net/

・https://checkio.org/

・https://paiza.jp/