Fastcampus

컴퓨터공학 입문 스쿨

Python Basic_Day5

2017.3.31

Code like PYTHON

Important Python Enhance Proposal

Layout

- 들여쓰기: 공백 4칸 or 탭(섞어쓰면 안됨)
- 한 줄은 79자(120자도 상관없음)
- 클래스정의와 최상위 함수는 두 줄을 띄움
- 클래스 내 메소드는 한 줄을 띄움

Important Python Enhance Proposal

Variables

- `_variable` : 내부적으로 사용되는 변수
- `print_` : 파이썬 키워드와 충돌 방지

Naming Convention

- 클래스 이름은 `CamelCase`
- 함수, 변수, 메소드 이름은 `snake_case`

파이썬에서 쓰이지 않는 네이밍 규칙

- `chHungarianNotation`
- `javaScriptStyleCamelCase`

Programming pattern

1세대

- 0,1로 이루어진 Low Level 언어(기계어)

```
10101111 00011010 01010100
```

2세대

- 1:1 대응 기호를 활용(어셈블리어)

```
push 1
add esp, 4
xor eax, eax
```

Programming pattern

3세대

- 프로시져(절차적 프로그래밍(x)) - Fortran
  프로시져(루틴, 서브루틴, 매소드, 함수)를 이용
  ~~GOTO~~

```
  READ INPUT TAPE 5, 501, IA, IB, IC
501 FORMAT (I5)
  IF (IA) 777, 777, 701
701 IF (IB) 777, 777, 702
702 IF (IC) 777, 777, 703
703 IF (IA+IB-IC) 777,777,704
704 IF (IA+IC-IB) 777,777,705
705 IF (IB+IC-IA) 777,777,799
777 STOP 1
```

Programming pattern

- 구조적 프로그래밍(Top-down) - C, Pascal
  단계단계 밟으며 문제를 해결하도록 작성

```c
#include <stdio.h>

int main(void)
{
    printf("hello, world\n");
}
```

Programming pattern

4세대

- 객체지향(Bottom-up) - C++, java, python
  데이터와 데이터를 처리할 메소드를 객체로 구성하고, 이를 조합하여 사용

Programming pattern

5세대

- 함수형 - Scala, Haskel
  자료처리를 수학적인 계산으로 인식하고 상태와 변수등의 개념을 최소화
  (명령형: 상태를 바꾸는 것, 함수형: 함수의 응용)

Toggl

https://blog.toggl.com/wp-content/uploads/2016/12/toggl-it-jobs-explained-with-changing-lightbulb.jpg

https://assets.toggl.com/images/toggl-how-to-save-the-princess-in-8-programming-languages.jpg

Ethiopian Multiplication

2로 나누고 곱하는 과정으로 두 수의 곱을 구현하는 방법

https://en.wikipedia.org/wiki/Ancient_Egyptian_multiplication

```
12 *     7       struck  ---
 6      14       struck  ---
 3      28       keep     28
 1      56       keep     56
--> 28 + 56 = 84
```

Recursive

```
times = int(input("How many times want to curse the beast??: "))
def recurse_beast(a):
        if a == 0:
                print("curse complete!")
        else:
                print("Fusion!!!(%d times left)" % a)
                recurse_beast(a-1)
```

List Comprehension

존재하는 리스트를 활용하여 새로운 리스트를 생성하는 방법

비슷한 표현들

- Set Comprehension
- Dictionary Comprehension
- Parallel list Comprehension

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = []
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

List Comprehension

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = []
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list]
```

List Comprehension - another example

```
old_list = [1, 2, 3, 4, 5,]

doubled_list = []
for i in old_list:
        if i % 2 == 0:
                doubled_list.append(i * 2)
```

```
doubled_list = [i * 2 for i in old_list if i % 2 == 0]
```

File I/O

File I/O

```
f = open(filename, mode)
f.close()
```

mode

r - 읽기모드

w - 쓰기모드

a - 추가모드(파일의 마지막에 새로운 내용을 추가)

Create New File

```
f = open("Newfile.txt", 'w')
f.close()
```

Write text

```
f = open("Newfile.txt", 'a')
for i in range(1,11):
        text = "line %d. \n" % i
    f.write(text)
f.close()
```

Read text

```
f = open("Newfile.txt", 'r')
text = f.readline()
print(text)
f.close()
```

Read All text

```
f = open("Newfile.txt", 'r')
while True:
        text = f.readline()
        if not text: break
        print(text)
f.close()
```

Read All text using readlines

```
f = open("Newfile.txt", 'r')
texts = f.readlines()
for text in texts:
        print(texts)
f.close()
```

Add text

```
f = open("Newfile.txt", 'a')
for i in range(11, 20):
        text = "New line %d \n" % i
        f.write(text)
f.close()
```

Get rid of f.close()

```
with open("foo.txt", 'w') as f:
        f.write("foo is text dummy")
```

Error Handle

by using `try, except`

필요한 만큼만 적절히 사용하셔야 합니다 by PEP 8

Error Handle - Syntax

```
try:
        실행문
except:
        실행문
```

Error Handle - ValueError

```
try:
        some_input = int(input("type some number: "))
except ValueError:
        print("I said type some NUMBER!!!!")
```

Error Handle - ValueError

```
try:
        some_input = int(input("type some number: "))
except ValueError as e:
        print("I said type some NUMBER!!!!")
    print(e)
```

Error Handle - FileNotFoundError

```
try:
        f = open('error_example.txt', 'r')
except FileNotFoundError as e:
        print(e)
else:
        text = f.read()
    f.close()
```

Error Handle - Multiple Error

```
try:
        ...
except error type 1:
        ...
except error type 2:
        ...
```

Error Handle - Pass Error

```
try:
        f = open('error_example.txt', 'r')
except FileNotFoundError as e:
        pass
else:
        text = f.read()
    f.close()
```