

Pylearn 1

Quick Start

2021.09

김윤경

openbase 

<https://secuwave.github.io/pylearn/index>

The subject comes first, the medium second.

주제가 제일 중요하고 도구는 그 다음이다.

- Richard Prince

스크립트는 도구이다.

스크립트는, 할 일의 목표와 과정을 정확히 이해하고,
프로세스가 결정된 후, 그 일을 효율적으로 수행하는데 사용하는 도구 중 하나이다.

Script는 로 실행한다.

스크립트는 일종의 대본이다.

목표를 달성하기 위한 작업 프로세스를 명령 해석기의 문법에 맞춰 정의한 것이 스크립트이고, 해석기를 인터프리터(Interpreter)라고 한다.
인터프리터는 스크립트에 정의된 일련의 과정을 하나씩 해석하여 실행한다.

- (1) 특정 OS의 명령어를 해석하여 수행하는 인터프리터: Bash, Windows 파워 셸
- (2) 프로그래밍 문법을 갖춘 전용 인터프리터: Python, Ruby, JavaScript, TCL

* 인터프리터의 속성, 문법 또는 명령어세트를 기본적으로 익혀야 스크립트 활용 가능

스크립트	확장자	인터프리터	시스템
Windows Batch	.bat	COMMAND.COM cmd.Exe	DOS Windows
Linux Shell	.sh	bash, csh, ksh...	LINUX 계열 OS (Red Hat, CentOS, Ubuntu, BSD)
Python	.py	Python	-
Java Script	.js	브라우저	-


자바스크립트의 인터프리터는 브라우저
브라우저가 html과 자바스크립트를 해석/실행

```
<!doctype html>
<html>
<head>
  <title>Test</title>
</head>
<body>
  <h1>greeting</h1>
</body>
<script>
  birthday_member = "오픈베이스"
  member = prompt("이름을 입력하세요: ");
  if (member == birthday_member) {
    alert(member + "님 생일축하합니다!");
  }
  else {
    alert(member + "님 반갑습니다!");
  }
</script>
</html>
```

Java Script

html에서 자바스크립트 부분을 greeting.js 파일로 분리 저장해도 같음

```
<!doctype html>
<html>
<head>
  <title>Test</title>
</head>
<body>
  <h1>greeting</h1>
</body>
<script src="greeting.js"></script>
<!--
<script>
  birthday_member = "오픈베이스"
  member = prompt("이름을 입력하세요: ");
  if (member == birthday_member) {
    alert(member + "님 생일축하합니다!");
  }
  else {
    alert(member + "님 반갑습니다!");
  }
</script>
-- >
</html>
```



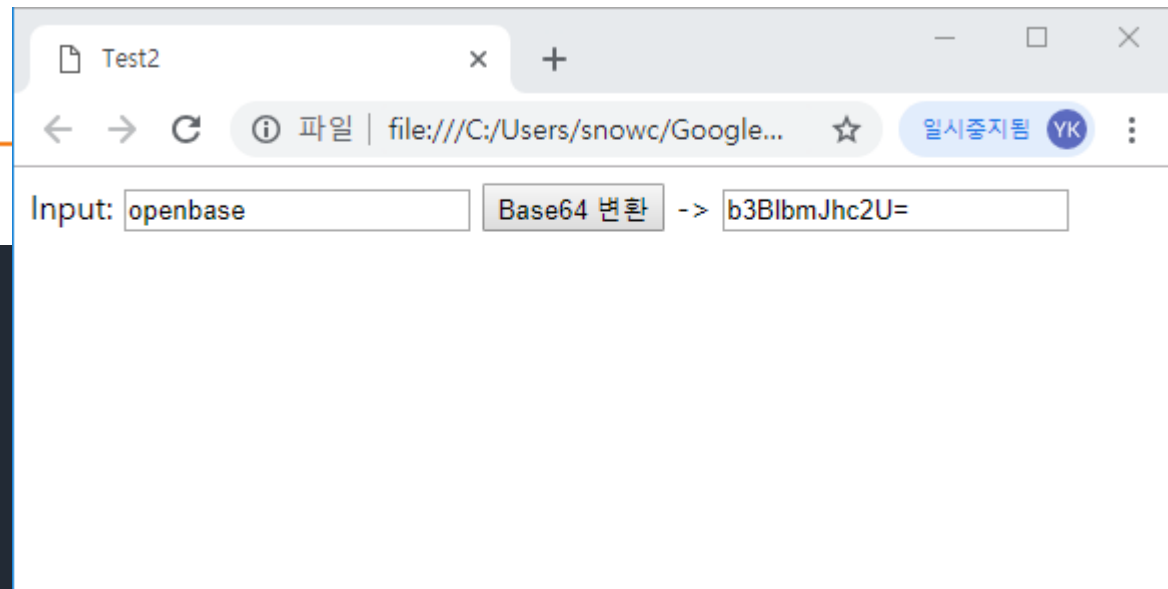
```
birthday_member = "오픈베이스"
member = prompt("이름을 입력하세요: ");
if (member == birthday_member) {
  alert(member + "님 생일축하합니다!");
}
else {
  alert(member + "님 반갑습니다!");
}
```

자바스크립트만 별도 파일에 분리하고 src 속성에 파일 이름을 지정하여 정의할 수 있습니다.
<!-- --> 은 html의 주석입니다.

Java Script — Base64 변환기

자바스크립트를 이용한 '평문 -> Base64' 변환기

```
<!doctype html>
<html>
  <head>
    <title>Test2</title>
  </head>
  <body>
    <form id="testform">
      Input: <input type="text" name="user_input">
      <input type="button" value="Base64 변환" onclick="convert2Base64()"></input>
      ->
      <input type="text" name="base64">
    </form>
  </body>
  <script type="text/javascript">
    function convert2Base64()
    {
      result = window.btoa(document.forms["testform"]["user_input"].value)
      document.forms["testform"]["base64"].value = result;
    }
  </script>
</html>
```



브라우저에는 btoa()라는 Base64 변환 함수가 내장
돼 있음
html form을 이용해서 문자열을 입력 받아 자바스크
립트로 Base64 변환

Linux Shell — 기본 셸 Bash

리눅스는 GUI로도 액세스하지만 CLI(Command Line Interpreter)를 많이 쓰기 때문에 셸 스크립트가 중요

```
$ cat /etc/shells # 리눅스에서 사용 가능한 셸
```

```
$ echo $SHELL # 현재 셸 확인
```

여러 리눅스 셸(인터프리터) 중 기본 셸은 bash(Born Shell)

보통 리눅스에 로그인하면 default로 bash 셸을 하나 얻게 됨

리눅스 셸은 명령어에 즉시 반응하는 인터프리터

```
$ a=1
$ b=100

$ ret1=$a+$b
$ echo $ret1
???
$ ret2=`expr $a + $b`
$ echo $ret2
???
$ let ret3=$a+$b
$ echo $ret3
???
```

변수에 값을 할당할 때 '=' 앞뒤에 space있으면 안됨
a = 1 (X)
정의된 변수를 사용할 때는 '\$'를 변수명 앞에 붙임

가정: 서비스(포트 5601) 연결이 불규칙하게 자주 끊어지는데 메모리 사용량과 관련이 있는 것 같아서 연결상태와 메모리 사용량을 같이 체크할 필요가 있다.

리눅스 명령:

- `Date`: 시스템 시간 확인. 체크 시간을 표시
- `netstat -an | grep 5601`: 포트 5601 상태 보기
- `vmstat`: 시스템 메모리 사용 확인

적용: 매번 체크할 때 명령어들을 반복 실행하므로 bash 스크립트 파일로 만듦

```
#!/bin/bash

date
netstat -an | grep 5601
vmstat
echo -----
```

파일에 실행 퍼미션을 주고 실행

```
$ chmod 755 my_script.sh
$ ./my_script.sh
```

24시간 모니터링을 하려면?

언제 문제가 생길지 모르니 포트 연결상태와 메모리 사용량을 3초마다 체크.

- 구조: while [유지 조건] do 실행할 일들 done
- 한 회차의 체크 후 다음 체크 사이에 3초 간격이 생기도록 sleep 추가

```
#!/bin/bash

while [ 1 ]
do
    date
    netstat -an | grep 5601
    vmstat
    echo -----
    sleep 3
done
```

#! (shebang line)

- = Hash(#) + Bang(!)
- 스크립트를 실행할 해석기를 지정
- 스크립트 파일 첫 줄에 정의

while loop 유지조건의 1은 'true'입니다. 그러므로 이 스크립트는 외부에서 중단시키기 전까지 실행됩니다.

- [1] (X)
- [1] (0)

3초마다 체크 결과를 화면에 찍지만 계속 화면만 볼 수 없음.

모니터링 정보를 파일에 기록

```
#!/bin/bash

while [ 1 ]
do
    date >> report.dat
    netstat -an | grep 5601 >> report.dat
    vmstat >> report.dat
    echo ----- >> report.dat
    sleep 3
done
```

> : 파일을 새로 만들어서 기록. 기존에 파일이 있으면 삭제하고 새로 만들.
>> : 파일의 끝에 내용을 추가 기록. 기존에 파일이 없으면 새 파일을 만들어서 기록.

데이터가 쌓이는 현황을 보려면:

```
$ tail -f ./report.dat
```