Shebang이란?

Namtae Kim

July 17, 2023

Contents

0.1 reference

- PEP 394: The “python” Command on Unix-Like Systems

– `https://peps.python.org/pep-0394/`

1 GOAL

python 프로그래밍을 하다보면, 파이썬 스크립트 최상단에 ”!/usr/bin/env python” 이라 적혀있는 코드를 볼 수 있다. 처음에는 생각없이 사용했었는데, conda 가상환경을 사용하면서 conda env에 설치된 interpreter에 맞게 수정해야 하는지 의문이 들었다. 그래서 이번 기회에 shebang이 무엇인지, shebang을 어떻게 사용하면 되는지 정리해보고자 한다.

2 Shebang 이란?

앞서 설명했듯이 ”!/usr/bin/env python” 와 같이, 파이썬 스크립트 최상단에 적힌 부분을 shebang이라 부른다. 먼저 shebang을 사용한 이유에 대해 알아보자. PEP 394에서는 shebang을 python 가상환경에서 ”유연하게” 코드를 배포하기 위해 사용하는 문법이라고 소개를 했다. 나는 처음에 ”/usr/bin/env” 만 보고,

```
(robostackenv) knt0613@Home:~$ env
SHELL=/bin/bash
ROS_VERSION=1
DEBUG_FFLAGS=-march=nocona -mtune=haswell -ftree-vectorize -fPIC -fstack-protect
or-strong -fno-plt -O2 -ffunction-sections -pipe -isystem /home/knt0613/minicond
a3/envs/robostackenv/include -march=nocona -mtune=haswell -ftree-vectorize -fPIC
 -fstack-protector-all -fno-plt -Og -g -Wall -Wextra -fcheck=all -fbacktrace -fi
mplicit-none -fvar-tracking-assignments -ffunction-sections -pipe
SESSION_MANAGER=local/Home:@/tmp/.ICE-unix/1716,unix/Home:/tmp/.ICE-unix/1716
```

Figure 1: env terminal

```
(robostackenv) knt0613@Home:~$ env python
Python 3.9.16 | packaged by conda-forge | (main, Feb  1 2023, 21:39:03)
[GCC 11.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Figure 2: env python

시스템에 종속적으로 python interpereter를 불러오고 가상환경 python interpreter를 불러오기 위해서는 따로 path를 지정해주어야 하는줄 알았다. 예를 들면 "/home/knt0613/miniconda3/env/bin/python" 처럼 말이다. 하지만 이렇게 지정할 경우, PEP에서 설명되었듯이 "유연하게" 코드를 배포하지 못하게 된다. 사실 알고보니 완전히 잘못된 생각이였다.

3 /usr/bin/env

알고보니 /usr/bin/env 자체가 바이너리 파일이였다. 터미널에 바로 "env" 만 입력하면 바이너리 파일이 실행된다. Fig. 1 가 "env" 를 커맨드한 결과인데, 결과를 보면 환경변수들이 출력된 것을 확인할 수 있다. 그렇다면 env python을 치면 어떻게 될까? Fig. 2 을 보면 현재 사용하고 있는 가상환경의 python interpreter가 실행 된것을 확인할 수 있다. 즉 /usr/bin/env는 현재 사용하고 있는 터미널(쉘) 환경변수를 참고해서 적절한 바이너리 파일을 실행하는 역할을 하는 것으로 이해하면 된다. 즉 env python 뿐만아니라 env {binary file} 과 같은 형식도 가능하다. 즉 Fig. 3 와 같이, "env ls" command도 가능하다. 그렇다면 현재 참조하고 있는 환경변수는 어디서 확인가능 할 까? 터미널에서 $PATH 로 조회할 수 있다. Fig. 4 를 보면, 여러개의 bin folder 가 출력된 것을 확인할 수 있는데, env 명령어는 $PATH에 있는 폴더들을 조회해서 커맨드와 일치하는 바이너리 파일이 있는지 확인하고, 존재한다면 그 바이너리 파일을 실행시킨다.

```
(robostackenv) knt0613@Home:~$ env ls
번역_정리      catkin_ws  Documents  miniconda3  SLAM_framework  Study  Videos
2022-alphacar  Desktop    Downloads  SLAM        snap            test   Vision
```

Figure 3: env ls

```
(robostackenv) knt0613@Home:~$ $PATH
bash: /home/knt0613/miniconda3/envs/robostackenv/bin:/usr/local/cuda-11.8/bin:/u
sr/local/cuda-11.8/bin:/home/knt0613/miniconda3/condabin:/usr/local/sbin:/usr/lo
cal/bin:/usr/sbin:/usr/bin:/sbin:/bin:/usr/games:/usr/local/games:/snap/bin:/usr
```

Figure 4: $PATH

```
(robostackenv) knt0613@Home:~$ cat test.py
#! /usr/bin/env python

import sys
print(sys.executable)
```

Figure 5: test.py

4 shebang의 필요성

그렇다면 해당 가상환경을 키고, "python –" 와 같이 파이썬 스크립트를 실행하면 되는게 아니냐하는 궁금증이 생길수도 있다. 이 궁금증을 해결하기 위해서는 shebang이 사실 python 만의 문법이 아니라, UNIX 계열 OS에서 인터프리터 기반 언어를 지원하기 위해 생긴 문법이라는 것을 알아야 한다. UNIX 계열 OS의 인터프리터 기반 언어라 ...., 딱 하나만 손 꼽아보자면 쉘스크립트가 대표적일 것이다. 터미널부터 시작해서 UNIX 계열 OS에서 가장 일반적으로 사용하는 언어일 텐데 Shebang은 이를 지원하기 위해 만들어졌다. 몇 가지 실험을 통해 그 이유를 자세히 알아보자.

Fig. 5 와 같이, 실행된 파이썬 인터프리터의 위치를 출력하는 아주 간단한 코드를 작성하였다. robostackenv 라는 콘다 가상환경에서 "python test.py" 로 파이썬을 실행하면 robostackenv 가상환경의 인터프리터 위치가 출력될 것이다. (Fig. 6) (usr/bin/env python 도 robostackenv 의 파이썬 인터프리터를 가리키고, 현재 파이썬 스크립트 실행도 robostackenv환경에서 하므로,,) 여기서 shebang을 /usr/bin/env python 이 아니라, 직접 파이썬 인터프리터 위치를 설정하여도 실행되는 인터프리터가 바뀌는지 궁금해졌다. 그래서 shebang을 conda base env 의 파이썬으로 지정하고, 실행은 conda robostackenv env 의 파이썬으로 실행하였다. 그 결과 Fig. 8 처럼 경로가 바뀌지 않은채 실행 된것을 확인 할 수 있었다.

어찌 보면 당연한 결과인데, shebang은 "인터프리터 기반의 언어"를 지원하기 위해 만들어진 문법이다. 한 가지 예시 Fig. 9 를 살펴보자. 이 예시에서는 "./test" 커맨드로 test 파일을 실행하였다. 이처럼 어떤 인터프리터를 사용할지 지정하지 않고 스크립트를 실행할 경우, 시스템은 사용한 인터프리터를 찾게 되는데

```
(robostackenv) knt0613@Home:~$ python test.py
/home/knt0613/miniconda3/envs/robostackenv/bin/python
```

Figure 6: (robostackenv) python test.py

```
(robostackenv) knt0613@Home:~$ cat test.py
#! /home/knt0613/miniconda3/bin/python

import sys
print(sys.executable)
```

Figure 7: robostack 환경에서, base env의 python을 shebang으로 지정

```
(robostackenv) knt0613@Home:~$ python test.py
/home/knt0613/miniconda3/envs/robostackenv/bin/python
```

Figure 8: robostack 환경에서, base env의 python을 shebang으로 지정했을 때 결과

이때 사용되는 문법이 shebang이다. python – 커맨드로 사용할 경우 Shebang은 주석처리되어 인터프리터 선택에 영향을 주지 않게 되고, ./test처럼 인터프리터를 지정하지 않을 경우에만 사용된다. shebang은 여러 언어로 구성된 바이너리 파일을 연이어서 실행하는 자동화 프로그램을 만들때 유용하게 사용되는데, 예를 들어 서로 다른 가상환경 A, B 에서 실행되어야 하는 프로그램 a.py 와 b.py 를 연이어서 실행하도록 자동화 프로그램을 짜야 할때 사용하면 좋다. python interpreter를 자동화 스크립트에서 찾아서 배포할 필요없이, 파이썬 프로그램 상단에서 파이썬 인터프리터 위치를 지정해 놓고, 자동화 스크립트에서는 "./a ; ./b" 와 같은 형태로 실행하면 된다.

```
(robostackenv) knt0613@Home:~$ mv test.py  test
(robostackenv) knt0613@Home:~$ chmod +x test
(robostackenv) knt0613@Home:~$ ./test
/home/knt0613/miniconda3/bin/python
```

Figure 9: 바이너리 파일처럼 실행한 결과