aws re/start

시스템 관리에 Python 활용

Python 기본 사항

발표자 이름

날짜

© 2020, Amazon Web Services, Inc. 또는 자회사. All rights reserved.

학습 내용

강의 핵심 내용

학습 내용:

- 시스템 관리를 정의합니다.
- Python 함수로 사용자를 관리합니다.
- Python 코드에서 패키지를 처리합니다.
- os.system()과 subprocess.run()을 사용하여 Python에서 Bash 명령을 실행합니다.

시스템 관리란?

SysAdmin이라고도 합니다.

하드웨어와 소프트웨어 시스템을 관리하는 일입니다.

컴퓨터 시스템과 모든 관련 서비스가 잘 작동하도록 합니다.

다음과 같은 태스크가 포함됩니다.

- 새 하드웨어 또는 소프트웨어 설치
- 사용자 계정 생성 및 관리
- 서버 및 데이터베이스 등 컴퓨터 시스템 유지 관리
- 시스템 중단 등 다양한 문제에 대비하고 적절히 대응

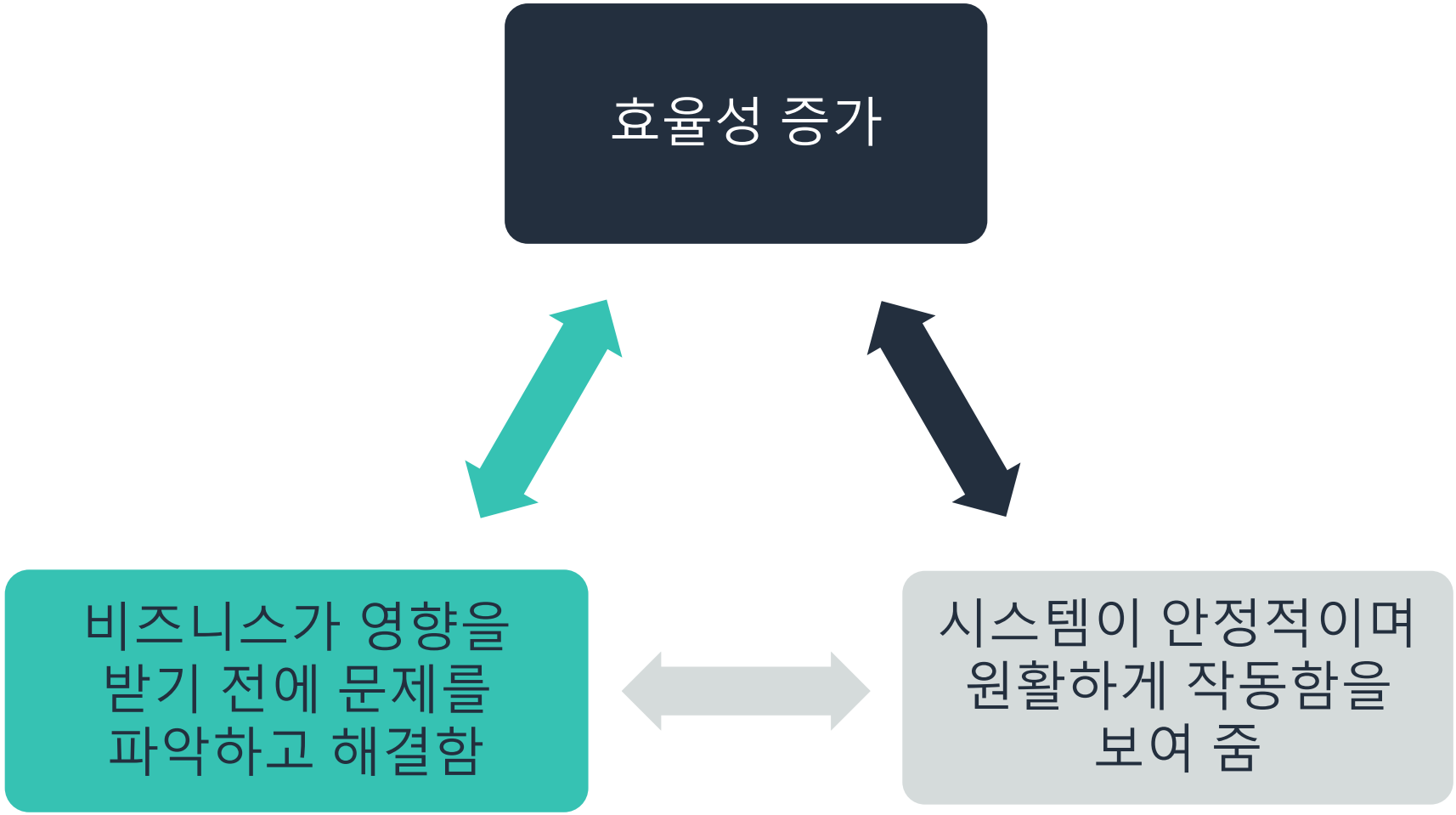
토론 질문

- Python을 사용하여 시스템 관리 태스크를 보다 수월하게 진행하는 방법은 무엇입니까?

© 2020, Amazon Web Services, Inc. 또는 자회사. All rights reserved.

aws re/start

SysAdmin의 이점



aws re/start

활동: 사용자와 협업

다음 슬라이드에는 사용자 관리를 위한 코드 조각이 포함되어 있습니다. Python 프로그래밍에 관해 습득한 지식을 활용하여 코드를 읽고 의미를 해독해 보십시오.

© 2020, Amazon Web Services, Inc. 또는 자회사. All rights reserved.

활동: 사용자 추가

```
def new_user():
    confirm = "N"
    while confirm != "Y":
        username = input("Enter the name of the user to add: ")
        print("Use the username '" + username + "'? (Y/N)")
        confirm = input().upper()
    os.system("sudo adduser " + username)
```

활동: 사용자 추가 - 솔루션

사용자가 Y를 입력할 때까지 계속됨

사용자 입력을 받아 변수에 할당

```
def new_user():
    confirm = "N"
    while confirm != "Y":
        username = input("Enter the name of the user to add: ")
        print("Use the username '" + username + "'? (Y/N)")
        confirm = input().upper()
    os.system("sudo adduser " + username)
```

while 루프가 종료된 후 사용자 이름으로 제공된 변수와 함께 Linux 명령 **sudo adduser** 호출

aws re/start

활동: 사용자 제거

```
def remove_user():
    confirm = "N"
    while confirm != "Y":
        username = input("Enter the name of the user to remove: ")
        print("Remove the user : '" + username + "'? (Y/N)")
        confirm = input().upper()
    os.system("sudo userdel -r " + username)
```


aws re/start

활동: 사용자 제거 - 솔루션

사용자가 Y를 입력할 때까지 계속됨

사용자 입력을 받아 변수에 할당

```
def remove_user():
    confirm = "N"
    while confirm != "Y":
        username = input("Enter the name of the user to remove: ")
        print("Remove the user : '" + username + "'? (Y/N)")
        confirm = input().upper()
    os.system("sudo userdel -r " + username)
```

while 루프가 종료된 후 사용자 이름으로 제공된 변수와 함께 Linux 명령 sudo userdel -r 호출

aws re/start

활동: 그룹에 사용자 추가(1)

```
def add_user_to_group():
    username = input("Enter the name of the user that you want to add to a
    group: ")
    output = subprocess.Popen('groups', stdout=subprocess.PIPE).communicate()[0]
    print("Enter a list of groups to add the user to")
    print("The list should be separated by spaces, for example:\r\n group1 group2
    group3")
    print("The available groups are:\r\n " + output)
    chosenGroups = str(input("Groups: "))
```


aws re/start

활동: 그룹에 사용자 추가(1) - 솔루션

협업하려는 사용자의 이름

```
def add_user_to_group():
    username = input("Enter the name of the user that you want to add to a
    group: ")
    output = subprocess.Popen('groups', stdout=subprocess.PIPE).communicate()[0]
    print("Enter a list of groups to add the user to")
    print("The list should be separated by spaces, for example:\r\n group1 group2
    group3")
    print("The available groups are:\r\n " + output)
    chosenGroups = str(input("Groups: "))
```

사용자가 추가되어야 하는 그룹의 목록

groups 명령을 수행하고 결과를 변수에 저장하면 나중에 사용자가 선택할 수 있는 출력이 됨

활동: 그룹에 사용자 추가(2)

```
output = output.split(" ")
chosenGroups = chosenGroups.split(" ")
print("Add To:")
found = True
groupString = ""
```


aws re/start

활동: 그룹에 사용자 추가(2) - 솔루션

이전 섹션의 문자열을 어레이로 분할함

이전 섹션의 문자열을 어레이로 분할함

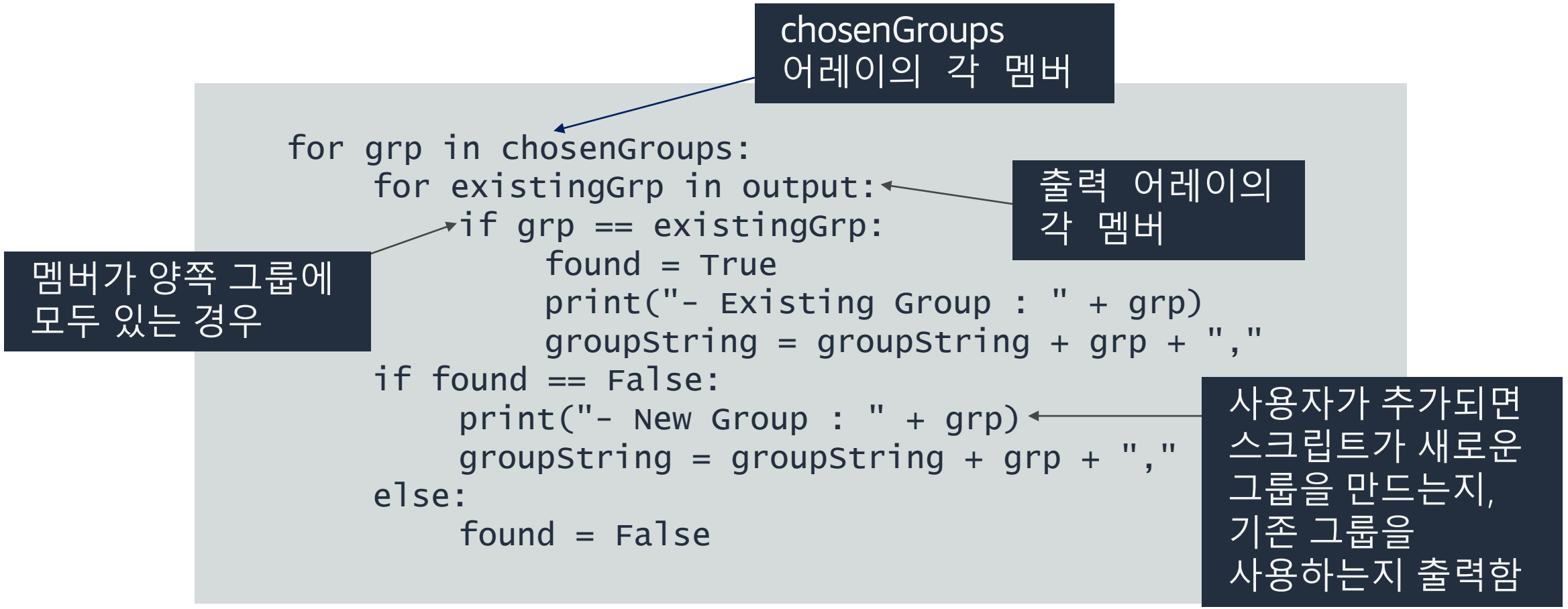
```
output = output.split(" ")
chosenGroups = chosenGroups.split(" ")
print("Add To:")
found = True
groupString = ""
```


aws re/start

활동: 그룹에 사용자 추가(3)

```
for grp in chosenGroups:
    for existingGrp in output:
        if grp == existingGrp:
            found = True
            print("- Existing Group : " + grp)
            groupString = groupString + grp + ","
    if found == False:
        print("- New Group : " + grp)
        groupString = groupString + grp + ","
    else:
        found = False
```

활동: 그룹에 사용자 추가(3) - 솔루션




```
for grp in chosenGroups:
    for existingGrp in output:
        if grp == existingGrp:
            found = True
            print("- Existing Group : " + grp)
            groupString = groupString + grp + ","
    if found == False:
        print("- New Group : " + grp)
        groupString = groupString + grp + ","
    else:
        found = False
```


aws re/start

활동: 그룹에 사용자 추가(4)

```
groupString = groupString[:-1] + " "
confirm = ""
while confirm != "Y" and confirm != "N" :
    print("Add user '" + username + "' to these groups? (Y/N)")
    confirm = input().upper()
if confirm == "N":
    print("User '" + username + "' not added")
elif confirm == "Y":
    os.system("sudo usermod -aG " + groupString + username)
    print("User '" + username + '" added)
```

활동: 그룹에 사용자 추가(4) - 솔루션

마지막 쉼표를 제거하고 줄 끝에 공백 추가

while 루프는 사용자가 Y 또는 N을 입력해야 종료됨

```
groupString = groupString[:-1] + " "
confirm = ""
while confirm != "Y" and confirm != "N" :
    print("Add user '" + username + "' to these groups? (Y/N)")
    confirm = input().upper()
if confirm == "N":
    print("User '" + username + "' not added")
elif confirm == "Y":
    os.system("sudo usermod -aG " + groupString + username)
    print("User '" + username + "' added)
```

사용자 입력을 받아 confirm 변수에 저장

while 루프가 종료되면 사용자가 Y 또는 N을 입력했는지 확인

앞에서 만든 그룹 및 사용자로 Linux 명령 **sudo usermod –aG** 호출

aws re/start

활동: 패키지 처리

다음 슬라이드에는 패키지 관리를 위한 코드 조각이 포함되어 있습니다. Python 프로그래밍과 관련해 습득한 지식을 활용하여 코드를 읽고 의미를 해독해 보십시오.

© 2020, Amazon Web Services, Inc. 또는 자회사. All rights reserved.

활동: 패키지 처리(1)

```
def install_or_remove_packages():
    iOrR = ""
    while iOrR != "I" and iOrR != "R":
        print("Would you like to install or remove packages? (I/R)")
        iOrR = input().upper()
    if iOrR == "I":
        iOrR = "install"
    elif iOrR == "R":
        iOrR = "remove"
```


aws re/start

활동: 패키지 처리(1) - 솔루션

```
def install_or_remove_packages():
    iOrR = ""
    while iOrR != "I" and iOrR != "R":
        print("Would you like to install or remove packages? (I/R)")
        iOrR = input().upper()
    if iOrR == "I":
        iOrR = "install"
    elif iOrR == "R":
        iOrR = "remove"
```

사용자가 패키지를 설치하려고 하는지, 제거하려고 하는지 확인

aws re/start

활동: 패키지 처리(2)

```
print("Enter a list of packages to install")
print("The list should be separated by spaces, for example:")
print(" package1 package2 package3")
print("Otherwise, input 'default' to " + iOrR + " the default packages listed in this program")
packages = input().lower()
if packages == "default":
      packages = defaultPackages
if iOrR == "install":
      os.system("sudo apt-get install " + packages)
```


aws re/start

활동: 패키지 처리(2) - 솔루션

입력의 형식 설명

```
print("Enter a list of packages to install")
print("The list should be separated by spaces, for example:")
print(" package1 package2 package3")
print("Otherwise, input 'default' to " + iOrR + " the default packages listed in this program")
packages = input().lower()
if packages == "default":
      packages = defaultPackages
if iOrR == "install":
      os.system("sudo apt-get install " + packages)
```

지정한 패키지와 함께 Linux 명령 **sudo apt-get install** 호출

사용자가 **default**로 지정하면 스크립트의 기본 패키지 목록 설치

aws re/start

활동: 패키지 처리(3)

```
elif iOrR == "remove":
    while True:
        print("Purge files after removing? (Y/N)")
        choice = input().upper()
        if choice == "Y":
            os.system("sudo apt-get --purge " + iOrR + " " + packages)
            break
        elif choice == "N":
            os.system("sudo apt-get " + iOrR + " " + packages)
            break
    os.system("sudo apt autoremove")
```


aws re/start

활동: 패키지 처리(3) - 솔루션

비교할 수 있도록 사용자 입력을 대문자로 변경

```
    elif iOrR == "remove":
        while True:
            print("Purge files after removing? (Y/N)")
            choice = input().upper()
            if choice == "Y":
                os.system("sudo apt-get --purge " + iOrR + " " + packages)
                break
            elif choice == "N":
                os.system("sudo apt-get " + iOrR + " " + packages)
                break
        os.system("sudo apt autoremove")
```

지정한 패키지와 함께 Linux 명령 **sudo apt-get –-purge remove** 호출

지정한 패키지와 함께 Linux 명령 **sudo apt-get remove** 호출

Linux 명령 **sudo apt autoremove**를 호출하여 오래된 패키지 파일이 있으면 제거

활동: 패키지 처리(4)

```
def clean_environment():
    os.system("sudo apt-get autoremove")
    os.system("sudo apt-get autoclean")
```

활동: 패키지 처리(4) - 솔루션

애플리케이션과 함께 설치되었으며 더 이상 시스템에서 사용되지 않는 종속 항목 제거

```
def clean_environment():
    os.system("sudo apt-get autoremove")
    os.system("sudo apt-get autoclean")
```

폐기된 deb-packages를 지움

이 Linux 명령 두 개를 함께 사용하면 환경을 최신 상태로 깔끔하게 유지하는 데 도움이 됨

aws re/start

활동: 패키지 처리(5)

```
def update_environment():
    os.system("sudo apt-get update")
    os.system("sudo apt-get upgrade")
    os.system("sudo apt-get dist-upgrade")
```

활동: 패키지 처리(5) - 솔루션

업그레이드해야 하는 패키지의 패키지 목록 업데이트, 최근 리포지토리에 추가된 새 패키지도 업데이트

```
def update_environment():
    os.system("sudo apt-get update")
    os.system("sudo apt-get upgrade")
    os.system("sudo apt-get dist-upgrade")
```

현재 OS 업데이트

참고: 이 명령은 OS를 상위 버전으로 업그레이드하지 않습니다. 예를 들어 Debian V8에서 명령을 실행해도 Debian V9로 업그레이드되지 않습니다.

설치된 모든 패키지의 업데이트를 다운로드 및 설치

aws re/start

더 나은 os.system(): subprocess.run()

Python V3에서 **os** 모듈은 지원이 중단되고 **subprocess** 모듈로 대체되었습니다.

모듈 지원 중단:

여전히 모듈을 사용할 수 있음

기존 스크립트에서 널리 사용되기 때문

그러나 같은 작업을 더 효과적으로 수행할 방법이 있음

`os.system()`의 등가 함수는 `subprocess.run()`입니다.

aws re/start

os.system()과 subprocess.run()의 비교

os.system()

- 주로 Linux의 Bash와 같은 하위 셸에서 실행됩니다.
- 셸은 주어진 문자열을 취하여 확장 문자를 해석합니다.
  - 예: `os.system("python -version")`

subprocess.run()

- 기본적으로 셸을 사용하지 않습니다. 대신 주어진 문자열을 이름으로 사용하여 프로그램을 실행하려고 시도합니다.
- 인수와 함께 명령을 실행하려면 목록을 전달해야 합니다.
  - 예: `subprocess.run(["python","-version"])`

subprocess.run()이 os.system()보다 더 나은 이유

다음과 같은 이유로 `subprocess.run()`이 `os.system()`보다 더 낫습니다.

안전	개발자가 실제 명령을 확인하지 않고 `os.system()`에 입력 문자열을 전달하는 경우가 자주 있습니다. 이는 위험한 일입니다. 예를 들어, 악의가 있는 사용자가 문자열을 전달해 파일을 삭제할 수 있습니다.
별도 프로세스	`subprocess.run()`은 Popen이라는 클래스로 구현되며, 이 클래스는 별도의 프로세스로 실행됩니다.
추가 기능	`subprocess.run()`은 실제로 Popen 클래스이기 때문에 `poll()`, `wait()`, `terminate()` 같은 유용한 메서드가 있습니다.

학습 내용 확인

시스템 관리란 무엇입니까?

시스템 관리와 관련된 일반적인 태스크 하나를 예로 들어 보십시오.

시스템 관리의 이점을 한 가지 예로 들어 보십시오.

요점

- 시스템 관리는 소프트웨어와 하드웨어 시스템을 관리하는 일입니다.
- 시스템 관리를 통해 효율성을 높이며, 문제를 빠르게 파악하고 해결할 수 있을 뿐만 아니라 시스템 안정성을 보장할 수 있습니다.
- Python에서는 복잡한 결정을 내리는 코드를 실행하고 `os.system()`과 `subprocess.run()`을 호출하여 시스템을 관리함으로써 시스템 관리 성능을 개선합니다.

© 2020, Amazon Web Services, Inc. 또는 자회사. All rights reserved.