데스크톱 자동화

데스크톱 자동화

1. 마우스, 키보드 자동화
2. 폴더 및 파일관리 자동화
3. 이미지 자동화

마우스, 키보드 자동화

마우스, 키보드 자동화

- pyautogui 라이브러리

자동화 된 마우스와 키보드의 조작을
가능하게 하는 라이브러리

마우스, 키보드 자동화

```
# 모니터 화면 크기 출력
pyautogui.size()

# 현재 마우스 위치 출력
pyautogui.position()

# 마우스를 좌표(50, 100)으로 이동
pyautogui.moveTo(50, 100)

# 마우스를 좌표(50, 100)으로 2초간 이동
pyautogui.moveTo(50, 100, 2)
```

마우스, 키보드 자동화

```
# 현재 마우스 위치에서 좌클릭
pyautogui.click()

# 마우스를 좌표 (50,100)으로 이동하여 좌클릭
pyautogui.click(50, 100)

# 현재 마우스 위치에서 더블좌클릭
pyautogui.doubleClick()

# 현재 마우스 위치에서 우클릭
pyautogui.rightClick()
```

마우스, 키보드 자동화

```
# Hello world!를 입력. 한글지원 X
pyautogui.write('Hello world!')

# 0.1초 간격으로 Hello world!를 한 문자씩 입력
pyautogui.write('Hello world!', interval=0.1)

# 엔터키 입력. (shift, ctrl, enter, esc등 특수키 입력)
pyautogui.press('enter')

# 두 개의 키를 동시에 입력
pyautogui.hotkey('ctrl', 'c')
```

마우스, 키보드 자동화

- pyperclip 라이브러리

  - 클립보드를 조작할 수 있도록 도와주는 파이썬 라이브러리

  - 클립보드에 복사한 내용을 읽거나, 클립보드에 내용을 복사하는 등의 작업을 수행

마우스, 키보드 자동화

```python
import pyperclip

# 문자열 복사
pyperclip.copy('Hello, Pyperclip!')

# 클립보드에 있는 문자열 읽기
text = pyperclip.paste()
print(text)
```

미니 실습

(1) windows 메뉴를 열고 메모장 실행
(2) 3초 대기
(3) “Hello, world” 텍스트 입력
(4) “Ctrl + S” 키를 눌러 파일 저장 대화상자 오픈
(5) 3초 대기
(6) 파일 이름 “miniproject_pyautogui” 입력후 저장
(7) 프로그램 종료

마우스, 키보드 자동화

미니 실습

```python
import pyautogui
import pyperclip
import time

# 메모장 프로그램 실행
pyautogui.press('win')
pyautogui.write('notepad')
pyautogui.press('enter')

# 3초 대기
time.sleep(3)

# "Hello, world!" 텍스트 입력
pyautogui.write('Hello, world!')

# "Ctrl + S" 키 입력하여 파일 저장 대화상자 열기
pyautogui.hotkey('ctrl', 's')

# 3초 대기
time.sleep(3)

# 파일 이름 입력 및 저장
file_path = 'example.txt'
pyperclip.copy(file_path)
pyautogui.hotkey('ctrl', 'v')
pyautogui.press('enter')

# 프로그램 종료
pyautogui.hotkey('alt', 'f4')
```

상대경로
절대경로

폴더 및 파일관리 자동화

폴더 및 파일관리 자동화

- os 라이브러리

```
# 현재 운영 체제의 이름
os.name

# 현재 작업 디렉터리
os.getcwd()

# 작업 디렉터리를 주어진 경로로 변경
os.chdir(path)

# 주어진 경로의 모든 파일과 디렉터리를 반환
# 만약 경로를 지정하지 않으면 현재 작업 디렉터리 반환
os.listdir(path=None)

# 주어진 경로에 새 디렉터리를 생성
os.mkdir(path)

# 주어진 경로에 필요한 모든 상위 디렉터리와 최종 디렉터리를 생성
os.makedirs(path)
```

폴더 및 파일관리 자동화

- os 라이브러리

```
# 파일 삭제
os.remove(path)

# 파일이나 디렉터리의 이름을 변경
os.rename(src, dst)

# 인자로 주어진 경로들을 조합하여 하나의 경로를 생성
os.path.join('path1', 'path2', ...)
path = os.path.join('path', 'to', 'your', 'file.txt')
```

폴더 및 파일관리 자동화

- glob 라이브러리

```
# 주어진 패턴과 일치하는 파일 및 디렉터리 목록을 반환.
glob.glob(pattern)

# recursive는 하위 디렉터리까지 검색하여 반환
glob.glob(pattern, recursive=True)

# 예시
txt_files = glob.glob('*.txt')
custom_pattern_files = glob.glob('[0-9]*.txt')
txt_files_recursive = glob.glob('**/*.txt', recursive=True)
```

폴더 및 파일관리 자동화

- glob 패턴

1. `*` (asterisk): 모든 문자와 일치합니다. 파일 또는 디렉터리 이름의 일부 또는 전체와 일치할 수 있습니다.
   예: `*.txt`는 모든 텍스트 파일과 일치합니다.
2. `?` (question mark): 하나의 문자와 일치합니다. 파일 또는 디렉터리 이름의 특정 위치에 있는 하나의 문자와 일치하는 경우에 사용합니다.
   예: `file?.txt`는 file1.txt, file2.txt 등과 일치합니다.
3. `[...]` (square brackets): 대괄호 안에 있는 문자 중 하나와 일치합니다. 대괄호 안에 문자 또는 문자 범위를 지정할 수 있습니다.
   예: `[a-zA-Z]*.txt`는 영문자로 시작하는 모든 텍스트 파일과 일치합니다.
4. `[!...]` or `[^...]` (square brackets with exclamation mark or caret): 대괄호 안에 있는 문자와 일치하지 않는 문자를 찾습니다.
   예: `[!0-9]*.txt` 또는 `[^0-9]*.txt`는 숫자로 시작하지 않는 모든 텍스트 파일과 일치합니다.
5. `**` (double asterisk): 재귀적으로 모든 하위 디렉터리와 일치합니다. 이 패턴을 사용하려면 `glob.glob()` 또는 `glob.iglob()` 함수의 `recursive` 매개변수를 `True`로 설정해야 합니다.
   예: `**/*.txt`는 모든 하위 디렉터리의 텍스트 파일과 일치합니다.

폴더 및 파일관리 자동화

- shutil 라이브러리

```
# 파일 복사. src는 원본 파일 경로/dst는 대상 디렉터리 또는 파일 경로
shutil.copy('source.txt', 'destination.txt')

# 파일을 복사하고 파일의 메타데이터도 복사
shutil.copy2('source.txt', 'destination.txt')

# 디렉터리와 그 내용을 재귀적으로 복사
shutil.copytree('source_directory', 'destination_directory')

# 파일 또는 디렉터리를 이동하거나 이름 변경
shutil.move('source.txt', 'destination.txt')

# 주어진 실행 파일(cmd)의 경로를 찾아 반환 실행 파일이 경로에 없으면 None을 반환
shutil.which(cmd)
print(shutil.which('python'))
```

미니 실습

- 원본 파일이 위치한 폴더 경로, 이동할 파일들의 이름 패턴, 이동할 폴더 경로를 지정합니다.
- 이동할 폴더가 없으면 생성합니다.
- glob 라이브러리를 사용하여 원본 폴더에서 파일 경로 리스트를 가져옵니다.
- shutil 라이브러리를 사용하여 파일을 이동합니다.
- 파일 이동이 완료되면 해당 파일의 경로와 이동된 경로를 출력합니다.

폴더 및 파일관리 자동화

미니 실습

```python
import os
import glob
import shutil

# 원본 파일이 위치한 폴더 경로
src_dir = '/path/to/source'

# 이동할 파일들의 이름 패턴
file_pattern = '*.txt'

# 이동할 폴더 경로
dst_dir = '/path/to/destination'

# 이동할 폴더가 없으면 생성
if not os.path.exists(dst_dir):
    os.mkdir(dst_dir)

# 파일 경로 리스트 가져오기
file_list = glob.glob(os.path.join(src_dir, file_pattern))

# 파일 이동
for file_path in file_list:
    shutil.move(file_path, dst_dir)
    print(f'{file_path} moved to {dst_dir}')
```

이미지 자동화

이미지 자동화

- pillow (PIL) 라이브러리

이미지 처리를 위한 라이브러리

이미지 자동화

- pillow (PIL) 라이브러리

```python
from PIL import Image

# 이미지 파일을 연 다음 Image 객체로 반환합니다.
image = Image.open('example.jpg')

# 이미지의 크기, 모드, 형식 등의 정보를 확인합니다.
print(image.size)
print(image.mode)
print(image.format)

# 이미지를 파일로 저장합니다.
image.save('example_output.jpg')

# 이미지의 크기를 변경합니다.
resized_image = image.resize((width, height))

# 이미지를 회전합니다.
rotated_image = image.rotate(90)

# 이미지의 일부 영역을 자릅니다.
cropped_image = image.crop((left, upper, right, lower))
```

이미지 자동화

- pillow (PIL) 라이브러리

```
# 이미지 병합
image1 = Image.open('example1.jpg')
image2 = Image.open('example2.jpg')
merged_image = Image.new('RGB', (width, height))
merged_image.paste(image1, (0, 0))
merged_image.paste(image2, (0, image1.height))
```

이미지 자동화

- pillow (PIL) 라이브러리

```python
from PIL import Image, ImageFilter

# 이미지 열기
image = Image.open('image.jpg')

# 이미지 필터 적용
# 모션 블러 필터
blurred = image.filter(ImageFilter.BLUR)

# 가우시안 블러 필터
blurred_gaussian = image.filter(ImageFilter.GaussianBlur(radius=2))

# 엠보싱 필터
embossed = image.filter(ImageFilter.EMBOSS)

# 샤프닝 필터
sharpened = image.filter(ImageFilter.SHARPEN)

# 가장자리 강조 필터
edge_enhanced = image.filter(ImageFilter.EDGE_ENHANCE)

# 가장자리 강조(더 강한) 필터
edge_enhanced_more = image.filter(ImageFilter.EDGE_ENHANCE_MORE)
```