

파이썬 기초~응용

『2과목 :』 빅데이터 분석 도구

2025.06.23-07.04(10일, 70시간)

Prepared by Daekyeong Kim

Ph.D.

1. 빅데이터 분석 도구 -소개
 - 빅데이터 분석 도구 개요
 - 실습 환경 구축 Building a Data Analysis and AI(Artificial Intelligence) practice environment
 - Python 작업 몇 가지 규칙
2. 빅데이터 분석 도구- fundamental
 - Python 계산, 변수와 자료형
 - Python 자료구조
 - Python's Statement
 - 함수
 - 객체와 클래스
 - Moduler과 패키지
 - 파이썬의 예외처리와 내장함수
 - 파이썬 외장함수
 - 라이브러리
 - Python 과 Library
3. 빅데이터 분석 도구- advanced
 - 파이썬 표준 입/출력
 - 웹-Flask 개발 환경 만들기 등
4. Self 점검
 - 필기 평가
 - 실기 평가

『2-3』 빅데이터 분석 도구 -advanced

- 파이썬 표준 입/출력
- 웹-Flask 개발 환경 만들기
- ...





1. Syllabus

학습목표

- 이 워크샵에서는 파이썬 표준 입출력에 대해 알 수 있습니다.
- 파이썬으로 문자열 출력과 관련한 형식화하는 방법에 대해 알 수 있습니다.

눈높이 체크

- 문자열 형식화를 알고 계신가요?
- 파이썬 표준 입출력을 알고 계신가요?



1. 표준 입출력

출력 결과

● 두 문장의 출력 결과

- 첫 번째 문장은 콤마(,) 로 문자열들을 구분하였고 두 번째 문장은 더하기(+) 기호로 문자열들을 합친 결과

구분	내용
실습환경	py3_10_basic
소스코드	<pre>print("Coffee", "Tea") print("Coffee" + "Tea")</pre>
결과값1	Coffee Tea CoffeeTea
비고	

separator

● 문자열들을 구분

구분	내용
실습환경	py3_10_basic
소스코드	<pre>print("Coffee", "Tea", sep=",") # 값들을 콤마(,) 로 구분 print("Coffee", "Tea", "Smoothie", sep=" vs ") # 값들을 " vs " 로 구분 print("Coffee", "Tea", sep="," , end="?") print("무엇이 더 맛있을까요?")</pre>
결과값1	Coffee,Tea Coffee vs Tea vs Smoothie Coffee,Tea?무엇이 더 맛있을까요?
비고	

1. 표준 입출력

출력 결과

● ljust() 와 rjust()

구분	내용
실습환경	py3_10_basic
소스코드	<pre>scores = {"국어":100, "수학":50, "영어":80} for subject, score in scores.items(): # key, value print(subject, score) scores = {"국어":100, "수학":50, "영어":80} for subject, score in scores.items(): print(subject.ljust(8), str(score).rjust(4), sep=":")</pre>
결과값1	국어 100 수학 50 영어 80
결과값2	국어 : 100 수학 : 50 영어 : 80
비고	

1. 표준 입출력

출력 결과

- `zfill()`
 - `zfill()` 은 함께 전달해주는 숫자만큼의 공간을 확보하고 그 공간을 zero로, 즉 0 으로 채워주는(fill) 동작을 한다.

구분	내용
실습환경	py3_10_basic
소스코드	<pre>for num in range(1, 21): # 1~20 까지의 숫자 print("대기번호 : " + str(num)) for num in range(1, 21): # 1~20 까지의 숫자 print("대기번호 : " + str(num).zfill(3))</pre>
결과값1	<pre>... 대기번호 : 16 대기번호 : 17 대기번호 : 18 대기번호 : 19 대기번호 : 20</pre>
결과값2	<pre>... 대기번호 : 016 대기번호 : 017 대기번호 : 018 대기번호 : 019 대기번호 : 020</pre>
비고	

2. 파이썬과 문자

파이썬으로 글자를 출력하기

- 파이썬과 같은 프로그래밍 언어에서는 글자를 문자열(string)이라고 부른다. 파이썬에서 문자열을 만들 때는 따옴표를 사용한다.
 - 따옴표에는 큰 따옴표와 작은 따옴표가 있으며 시작 따옴표와 종료 따옴표만 같으면 어느 것을 사용해도 상관없다.
 - 문자열을 출력하려면 print 명령을 사용한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("Hello!") print('한글도 쓸 수 있어요.')</pre>
결과값1	Hello! 한글도 쓸 수 있어요.
비고	

2. 파이썬과 문자

문자열 연산

- 문자열도 숫자처럼 덧셈과 곱셈 연산을 할 수 있다. 덧셈 연산은 두 문자열을 붙이고 곱셈 연산은 문자열을 반복한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("내 이름은 " + "홍길동" + "입니다.")</pre>
결과값1	내 이름은 홍길동입니다.
비고	

숫자를 문자열로 바꾸기

- 숫자를 문자열과 더하려면 str 명령을 써서 숫자를 문자열 자료형으로 바꾸어야 한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("*" * 10) n = 10 print("별표를 " + str(n) + "번 출력합니다.") print("*" * n)</pre>
결과값1	<pre>***** 별표를 10번 출력합니다. *****</pre>

비고

2. 파이썬과 문자

한 줄 띄우기

- print 명령은 한 번 호출할 때마다 한 줄씩 출력한다. 만약 print 명령을 한 번만 쓰면서 여러 줄에 걸쳐 출력을 하고 싶으면 문자열에 "다음 줄 넘기기(line feed) 기호"인 `\n`를 넣어야 한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("한 줄 쓰고\n그 다음 줄을 쓴다.")</pre>
결과값1	한 줄 쓰고 그 다음 줄을 쓴다.
비고	

줄을 바꾸지 않고 이어서 출력하기

- 반대로 print 명령을 여러번 쓰면서 줄은 바꾸지 않고 싶다면 다음과 같이 print 명령에 `end=""`이라는 인수를 추가한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("한 줄 쓰고 ", end="") print("이어서 쓴다.")</pre>
결과값1	한 줄 쓰고 이어서 쓴다.
비고	

2. 파이썬과 문자

문자열 값을 가지는 변수

- 변수에는 숫자뿐만 아니라 문자열도 넣을 수 있다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>name = "홍길동" print("내 이름은 " + name + "입니다.") mark = "\$" n = 20 print(mark + " 기호를 " + str(n) + "번 출력합니다.") print(mark * n)</pre>
결과값1	<pre>내 이름은 홍길동입니다. \$ 기호를 20번 출력합니다. \$</pre>
비고	



2. 파이썬과 문자

따옴표를 출력하기

- 파이썬에서 두 가지 종류의 다른 따옴표를 쓸 수 있는 이유는 문자열이 따옴표를 포함하는 경우가 있기 때문이다. 만약 따옴표로 둘러싸인 문자열에 따옴표가 포함되어 있다면 파이썬은 그 부분에서 문자열이 끝난다고 인식하여 오류가 발생한다. 이처럼 문자열 안에 큰따옴표가 있어야 할 때는 전체 문자열을 작은따옴표로 둘러싸면 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print('둘리가 "호이!"하고 말했어요.') print("둘리가 '이제 어디로 가지?'하고 생각했어요.") # print("둘리가 "호이!"하고 말했어요") print("둘리가 \"호이!\"하고 말했어요")</pre>
결과값1	<pre>둘리가 "호이!"하고 말했어요. 둘리가 '이제 어디로 가지?'하고 생각했어요. 둘리가 "호이!"하고 말했어요</pre>
비고	

2. 파이썬과 문자

따옴표를 출력하기

- \" 사용하기

```
print("둘리가 \"호미!\"하고 말했어요")
```

Cell In[29], line 1

```
print("둘리가 \"호미!\"하고 말했어요")
```

SyntaxError: invalid syntax

```
print("둘리가 #\"호미!#\"하고 말했어요")
```

둘리가 "호미!"하고 말했어요



2. 파이썬과 문자

탈출문자

- 줄 바꿈 문자인 '\n' 삽입

코드	설명
① \n	개행 (줄바꿈)
② \v	수직 탭
③ \t	수평 탭
④ \r	캐리지 리턴
⑤ \f	폼 피드
⑥ \a	벨 소리
⑦ \b	백 스페이스
⑧ \000	널문자
⑨ \\	문자 "\"
⑩ \'	단일 인용부호(')
⑪ \"	이중 인용부호(")



2. 파이썬과 문자

탈출문자

- 줄 바꿈 문자인 '\n' 삽입

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("Life is short, You need Python.") print("Life is short,\n You need Python.") print("Pine \"Apple\"입니다.") # Pine"Apple"입니다. print("C:\\DEV\\PycharmProjects\\python_works") print("Red Apple\rPine") # PineApple print("Redd\bApple") # RedApple print("Red\tApple") # Red Apple</pre>
결과값1	<pre>Life is short, You need Python. Life is short, You need Python. Pine "Apple"입니다. C:\DEV\PycharmProjects\python_works PineApple RedApple Red Apple</pre>
비고	

2. 파이썬과 문자

여러 줄의 문자열 출력하기

- 파이썬에서 여러 줄의 문자열을 출력하거나 변수에 할당하려면, "문자" 나 '문자' 대신 `"""` 여러 줄의 문자열 `"""` 혹은 `"""여러 줄의 문자열"""` 을 사용하면 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>multi_line_string = """ 파이썬(영어: Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 <Monty Python's Flying Circus>에서 따온 것이다.""" print(multi_line_string)</pre>
결과값1	<pre>파이썬(영어: Python)은 1991년 프로그래머인 귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로, 플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑(dynamically typed) 대화형 언어이다. 파이썬이라는 이름은 귀도가 좋아하는 코미디 <Monty Python's Flying Circus>에서 따온 것이다.</pre>
비고	

2. 파이썬과 문자

문자열 치환

- 문자열에서 특정 문자를 다른 문자로 바꾸려면 `replace` 메서드를 사용한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<code>print("2023.03.01".replace(".", "-"))</code>
결과값1	2023-03-01
비고	

- 문자열의 공백을 없애려면 " " 공백 문자열을 "" 빈 문자열로 바꾸면 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<code>print("word with space".replace(" ", ""))</code>
결과값1	wordwithspac
비고	

2. 파이썬과 문자

문자열 처리 함수 종류

함수이름	의미
<code>str.upper()</code>	문자열 <code>str</code> 을 모두 대문자로 바꾸어 준다.
<code>str.count(x)</code>	문자열 <code>str</code> 중 <code>x</code> 와 일치하는 것의 개수를 반환한다.
<code>str.find(x)</code>	문자열 <code>str</code> 중 문자 <code>x</code> 가 처음으로 나온 위치를 반환한다. 없으면 <code>-1</code> 을 반환한다.
<code>str.index(x)</code>	문자열 <code>str</code> 중 문자 <code>x</code> 가 처음으로 나온 위치를 반환한다. 없으면 에러를 발생시킨다.
<code>str.join(s)</code>	<code>s</code> 라는 문자열의 각각의 요소 문자 사이에 문자열 <code>str</code> 를 삽입한다.
<code>str.lower()</code>	문자열 <code>str</code> 을 모두 소문자로 바꾸어 준다.
<code>str.lstrip()</code>	문자열 <code>str</code> 의 왼쪽 공백을 모두 지운다.
<code>str.rstrip()</code>	문자열 <code>str</code> 의 오른쪽 공백을 모두 지운다.
<code>str.strip()</code>	문자열 <code>str</code> 의 양쪽 공백을 모두 지운다.
<code>str.replace(s,r)</code>	문자열 <code>str</code> 의 <code>s</code> 라는 문자열을 <code>r</code> 이라는 문자열로 치환한다.
<code>str.split(s)</code>	문자열 <code>str</code> 을 <code>s</code> 라는 문자열로 나누어 리스트 값을 돌려준다.
<code>str.split()</code>	<code>s</code> 를 주지 않으면 공백으로 나누어 리스트 값을 돌려준다.
<code>str.swapcase()</code>	문자열 <code>str</code> 의 대문자는 소문자로, 소문자는 대문자로 각각바꾸어 준다.

2. 파이썬과 문자

문자열 처리 함수 사용방법

파일

소스코드

실습환경

py3_10_basic

```
python = "Life is short, You need Python."
```

소스코드

```
print(python.lower())
print(python.upper())
print(python[0].isupper()) # True : 0 번째 인덱스의 값이 대문자인지 확인
print(len(python)) # 17 : 띄어쓰기를 포함한 문자열의 전체 길이 (length)
print(python.replace("Python", "Java"))
index = python.index("n") # 처음으로 발견된 n 의 인덱스
print(index) # 5 : Python 의 n
index = python.index("n", index + 1) # 6 번째 인덱스 이후에 처음으로 발견된 n 의 인덱스
print(index)
find = python.find("n") # 처음으로 발견된 n 의 인덱스
print(find) # 5 : Python 의 n
find = python.find("n", find + 1) # 6 번째 인덱스 이후에 처음으로 발견된 n 의 인덱스
print(find)
#print(python.index("Java")) # Java 가 없기 때문에 에러가 발생하며 프로그램 종료
print(python.find("Java")) # Java 가 없으면 -1 을 반환(출력)하며 프로그램 계속 수행
print(python.count("n")) # 2 : 문자열 내에서 n 이 나온 횟수
```

결과값1

```
life is short, you need python.
LIFE IS SHORT, YOU NEED PYTHON.
True
31
Life is short, You need Java.
19
29
19
29
-1
2
```

비고

3. 문자열 형식화

% 기호를 사용한 문자열 형식화

- 파이썬에서는 복잡한 문자열 출력을 위한 문자열 형식화(string formatting)를 지원한다.
 - 문자열을 형식화하는 방법에는 % 기호를 사용한 방식과 format 메서드를 사용한 방식, 그리고 f 문자열을 사용하는 방식이 있다.
 - 문자열 뒤에 % 기호를 붙이고 그 뒤에 다른 값을 붙이면 뒤에 붙은 값이 문자열 안으로 들어간다. "문자열" % 값
 - 이 때 문자열의 어느 위치에 값이 들어가는지를 표시하기 위해 문자열 안에 % 기호로 시작하는 형식지정 문자열(format specification string)을 붙인다. 대표적인 형식지정 문자열은 다음과 같다.

형식지정 문자열	의미
%s	문자열 (String)
%c	문자 한개(character)
%d	정수 (Integer)
%f	부동소수 (floating-point)
%o	8진수
%x	16진수
%%	Literal % (문자 '%s' 자체)

3. 문자열 형식화

고급 형식지정 문자열

- 형식지정 문자열은 여러가지 숫자 인수를 가질 수도 있다. % 기호 다음에 오는 정수는 값이 인쇄될 때 차지하는 공간의 길이를 뜻한다. 만약 공간의 길이가 인쇄될 값보다 크면 정수가 양수일 때는 값을 뒤로 보내고 공백을 앞에 채우거나 반대로 정수가 음수이면 값을 앞으로 보내고 공백을 뒤에 채운다. 만약 % 기호 다음에 소숫점이 있는 숫자가 오면 점 뒤의 숫자는 실수를 인쇄할 때 소숫점 아래로 그만큼의 숫자만 인쇄하라는 뜻이다.

고급 형식지정 문자열	의미
%20s	전체 20칸을 차지하는 문자열(공백을 앞에 붙인다.)
%-10d	전체 10칸을 차지하는 숫자(공백을 뒤에 붙인다.)
%.5f	부동소수점의 소수점 아래 5자리까지 표시

3. 문자열 형식화

고급 형식지정 문자열

파일	소스코드
실습환경	py3_10_basic # 방법 1
소스코드	<pre>print("나는 %d살입니다." % 20) # 나는 20살입니다 print("나는 %s을 좋아합니다." % "파이썬") # 나는 파이썬을 좋아합니다. print("Apple 은 %c로 시작해요." % "A") # Apple 은 A로 시작해요. print("나는 %s살입니다." % 20) # 나는 20살입니다 (%s 로도 정수값 표현 가능) print("나는 %s색과 %s색을 좋아해요." % ("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요.</pre>
결과값1	<pre>나는 20살입니다. 나는 파이썬을 좋아합니다. Apple 은 A로 시작해요. 나는 20살입니다. 나는 파란색과 빨간색을 좋아해요.</pre>
비고	

3. 문자열 형식화

format 메서드를 사용한 문자열 형식화

- 문자열 내에 중괄호 { } 를 집어 넣고 뒤에서 .format(값1, 값2, ...) 을 입력하면 이 값들이 문자열 내의 중괄호 부분에 들어 가게 된다. 이 때 { } 만 넣으면 순서대로 값1, 값2, ... 가 들어가게 되며 만약 {0}, {1} 과 같이 인덱스 값을 의미하는 숫자를 넣게 되면 {0} 위치에는 값1, {1} 위치에는 값2, ... 이런 식으로 들어 가게 된다.

파일	소스코드
실습환경	py3_10_basic # 방법 2
소스코드	<pre>print("나는 {}살입니다.".format(20)) # 나는 20살입니다. print("나는 {}색과 {}색을 좋아해요.".format("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요 print("나는 {0}색과 {1}색을 좋아해요.".format("파란", "빨간")) # 나는 파란색과 빨간색을 좋아해요 print("나는 {1}색과 {0}색을 좋아해요.".format("파란", "빨간")) # 나는 빨간색과 파란색을 좋아해요</pre>
결과값1	<pre>나는 20살입니다. 나는 파란색과 빨간색을 좋아해요. 나는 파란색과 빨간색을 좋아해요. 나는 빨간색과 파란색을 좋아해요.</pre>
비고	

3. 문자열 형식화

format 메서드를 사용한 문자열 형식화

- 문자열 내에 {이름} 과 같이 넣어두고, 마치 변수를 사용하는 것처럼 .format 내에서 이름과 값을 정의해두면, 그 이름에 해당하는 부분에 값을 집어넣게 된다.

파일	소스코드
실습환경	py3_10_basic # 방법 3
소스코드	<pre>print("나는 {age}살이며, {color}색을 좋아해요.".format(age=20, color="빨간")) # 나는 20살이며, 빨간색을 좋아해요 print("나는 {age}살이며, {color}색을 좋아해요.".format(color="빨간", age=20)) # 나는 20살이며, 빨간색을 좋아해요 (.format 뒤에 순서를 변경해도 괜찮아요)</pre>
결과값1	나는 20살이며, 빨간색을 좋아해요. 나는 20살이며, 빨간색을 좋아해요.
비고	

3. 문자열 형식화

format 메서드를 사용한 문자열 형식화

- format 방식에서 공백의 크기를 지정하거나 부동소수점의 소수점 아래 숫자를 지정할 때는 {}안에 : 기호를 넣고 그 뒤에 고급 형식지정 문자열을 넣는다. : 뒤에 오는 숫자는 공백의 크기를 뜻한다. <는 값을 왼쪽으로 붙이고 공백을 뒤로 붙인다. 반대로 >는 값을 오른쪽으로 붙이고 공백을 뒤로 붙인다. 소숫점의 자릿수를 지정할 때는 .(점)과 숫자, 그리고 f 글자를 사용한다. , 기호를 넣으면 영미권에서 숫자를 쓸 때 천(1000)단위마다 붙이는 쉼표(thousand comma)를 붙인다.

고급 형식지정 문자열	의미
{:>10}	전체 10칸을 차지하며 공백을 앞에 붙임 (문자열을 오른쪽에 붙여서 출력)
{:<10}	전체 10칸을 차지하며 공백을 뒤에 붙임 (문자열을 왼쪽에 붙여서 출력)
{:^10}	전체 10칸을 차지하며 공백을 앞뒤에 붙임 (문자열을 중앙에 붙여서 출력)
{:.5f}	부동소수점의 소수점 아래 5자리까지 표시
{:,}	천단위 쉼표 표시

3. 문자열 형식화

format 메서드를 사용한 문자열 형식화

예

구분	내용
실습환경	py3_10_basic
소스코드	<pre>print("[{:<20}]" .format("")) print("[{:>20}]" .format("")) print("[{: ^20}]" .format("")) print("[{:20.5f}]" .format(1 / 3)) print("[{:20,}]" .format(1234567890))</pre>
결과값1	<pre>[* [[* [0.33333] [1,234,567,890]</pre>
비고	

3. 문자열 형식화

format 메서드를 사용한 문자열 형식화

- 만약 > 기호앞에 문자열을 쓰면 해당 문자열로 공백을 채운다.

고급 형식지정 문자열	의미
{:*>10}	전체 10칸을 차지하며 "*"을 앞에 붙임 (문자열을 오른쪽에 붙여서 출력)
{:*<10}	전체 10칸을 차지하며 "*"을 뒤에 붙임 (문자열을 왼쪽에 붙여서 출력)
{:*^10}	전체 10칸을 차지하며 "*"을 앞뒤에 붙임 (문자열을 중앙에 붙여서 출력)

구분	내용
실습환경	py3_10_basic
소스코드	<pre>print("[{:<20}]" .format("*")) print("[{:>20}]" .format("*")) print("[{: ^20}]" .format("*"))</pre>
결과값1	<pre>[*-----] [-----*] [-----*-----]</pre>
비고	

3. 문자열 형식화

f 문자열

- 파이썬 3.6부터는 f 문자열(f-string)이라는 것을 사용할 수 있다. f 문자열은 문자열의 앞에 f 글자를 붙인 문자열이다. f 문자열에서는 {} 안에 변수의 이름을 바로 사용할 수 있다.

파일	소스코드
실습환경	py3_10_basic # 방법 4 (파이썬 버전 3.6 부터 가능)
소스코드	age = 20 color = "빨간" print(f"나는 {age}살이며, {color}색을 좋아해요.") # 나는 20살이며, 빨간색을 좋아해요.
결과값1	나는 20살이며, 빨간색을 좋아해요.
비고	

3. 문자열 형식화

f 문자열

- f 문자열에서 공백의 크기 등을 지정할 때는 format 방법과 같은 고급 형식지정 문자열을 사용할 수 있다.

구분	내용
실습환경	py3_10_basic
소스코드	<pre>number = 1234567 print(f"[{number:<20}]") print(f"[{number:>20}]") print(f"[{number:^20}]") print(f"[{number:-<20}]") print(f"[{number:->20}]") print(f"[{number:-^20}]") print(f"[{number:-^20,}]") f_number = 3.141592 print(f"[{f_number:20.3f}]")</pre>
결과값1	<pre>[1234567] [1234567] [1234567] [1234567-----] [-----1234567] [-----1234567-----] [-----1,234,567-----] [3.142]</pre>
비고	



4. 파일 입/출력

텍스트 파일 만들고 읽기

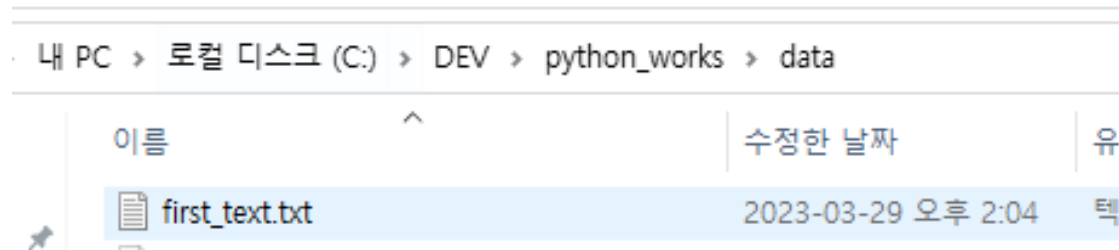
- 데이터는 거의 예외 없이 파일에 저장된다. 파일은 텍스트 파일, CSV 파일, Excel 파일 또는 기타 유형의 파일일 수 있다. 이러한 파일에 액세스하고 데이터를 읽는 방법을 알게 되면 Python에서 데이터를 처리, 조작 및 분석하는 도구를 얻을 수 있다. 초당 많은 파일을 처리할 수 있는 프로그램이 있는 경우, 각 작업을 일회성으로 수행하는 것보다 프로그램을 작성하는 것의 결과를 실제로 보게 된다. 스크립트가 처리하는 파일을 Python에 알려야 한다. 파일 이름을 프로그램에 하드코딩할 수 있지만 그렇게 하면 다양한 파일에서 프로그램을 사용하기가 어렵다. 파일에서 읽는 다양한 방법은 명령 프롬프트 또는 터미널 창의 명령줄에서 Python 스크립트 이름 뒤에 파일 경로를 포함하는 것이다.
- 이 방법을 사용하려면 스크립트 상단에 내장된 sys 모듈을 가져와야 한다. 스크립트에서 sys 모듈이 제공하는 모든 기능을 사용할 수 있도록 하려면 스크립트 맨 위에 import sys를 추가한다.



4. 파일 입/출력

단일 텍스트 파일 읽기

- 다음 처럼 텍스트 파일을 만들고 다음 6줄을 작성합니다.



파일	소스코드
실습환경	py3_10_basic
소스코드	I'm already much better at Python.
결과값1	
비고	



4. 파일 입/출력

단일 텍스트 파일 읽기

- `sys` 모듈을 가져오면 `argv` 목록 변수를 마음대로 사용할 수 있다. 이 변수는 명령줄 인수 목록(스크립트 이름을 포함하여 명령줄에 입력한 모든 것)을 캡처하여 Python 스크립트로 전달한다. 모든 목록과 마찬가지로 `argv`에는 인덱스가 있다. `argv[0]`은 스크립트 이름이다. `argv[1]`은 명령줄에서 스크립트에 전달된 첫 번째 추가 인수이며, 이 경우 `first_script.py`가 읽을 파일의 경로가 된다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>#!/usr/bin/env python3 from math import exp, log, sqrt import re from datetime import date, time, datetime, timedelta from operator import itemgetter import sys</pre>
결과값1	
비고	



4. 파일 입/출력

단일 텍스트 파일 읽기

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>print("Output ###: ") with open("datasets/first_text.txt", 'r', encoding='utf-8') as filereader: for row in filereader: print(row.strip()) filereader.close()</pre>
결과값1	<pre>Output ###: I'm already much better at Python.</pre>
비고	



4. 파일 입/출력

readline() 함수 이용하기

- `open("*.txt", 'r')`로 파일을 읽기 모드로 연 후 `readline()`을 사용해서 파일을 읽어 출력한다. `readlines` 함수는 파일의 모든 줄을 읽어서 각각의 줄을 요소로 갖는 리스트로 돌려준다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>F1 = open("datasets/first_text.txt", 'r', encoding='utf-8') line=F1.readline() print(line) F1.close()</pre>
결과값1	I'm
비고	



4. 파일 입/출력

readline() 함수 이용하기

- `open("*.txt", 'r')`로 파일을 읽기 모드로 연 후 `readline()`을 사용해서 파일을 읽어 출력한다. `readlines` 함수는 파일의 모든 줄을 읽어서 각각의 줄을 요소로 갖는 리스트로 돌려준다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>F1=open("datasets/first_text.txt", 'r', encoding='utf-8') while True: line=F1.readline() if not line: break print(line) F1.close()</pre>
결과값1	<pre>I'm already much better at Python.</pre>
비고	



4. 파일 입/출력

read 함수 사용하기

- read()는 파일의 내용 전체를 문자열로 돌려준다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>F2=open("datasets/first_text.txt", 'r', encoding='utf-8') line=F2.read() print(line) F2.close()</pre>
결과값1	<pre>I'm already much better at Python.</pre>
비고	



4. 파일 입/출력

추가 모드('a')

- 쓰기 모드('w')로 파일을 열 때 이미 존재하는 파일을 열면 그 파일의 내용이 모두 사라지게 된다. 하지만 원래 있던 값을 유지하면서 단지 새로운 값만 추가해야 할 경우도 있다.
- 이런 경우에는 파일을 추가 모드('a')로 열면 된다. *.txt 파일을 추가 모드('a')로 열고 write를 사용해서 결괏값을 기존 파일에 추가해 적는 예이다. 여기에서 추가 모드로 파일을 열었기 때문에 *.txt 파일이 원래 가지고 있던 내용 바로 다음부터 결과값을 적기 시작한다.

파일	소스코드	first_text.txt - Windows 메모장
실습환경	py3_10_basic	파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)
소스코드	<pre>F3=open("datasets/first_text.txt",'a') for i in range(11,20): line = "%d번째 줄입니다. \n" % i F3.write(line) F3.close()</pre>	<pre>I'm already much better at Python.11번째 줄입니다. 12번째 줄입니다. 13번째 줄입니다. 14번째 줄입니다. 15번째 줄입니다. 16번째 줄입니다. 17번째 줄입니다. 18번째 줄입니다. 19번째 줄입니다.</pre>
결과값1		
비고		



4. 파일 입/출력

추가 모드('a')

- 이런 경우에는 파일을 추가 모드('a')로 열면 된다. *.txt 파일을 추가 모드('a')로 열고 write를 사용해서 결괏값을 기존 파일에 추가해 적는 예이다. 여기에서 추가 모드로 파일을 열었기 때문에 *.txt 파일이 원래 가지고 있던 내용 바로 다음부터 결괏값을 적기 시작한다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>F4=open("datasets/first_text.txt", 'r', encoding='utf-8') line=F4.read() print(line) F4.close()</pre>
결과값1	<pre>I'm already much better at Python.11번째 줄입니다. 12번째 줄입니다. 13번째 줄입니다. 14번째 줄입니다. 15번째 줄입니다. 16번째 줄입니다. 17번째 줄입니다. 18번째 줄입니다. 19번째 줄입니다.</pre>
비고	

4. 파일 입/출력

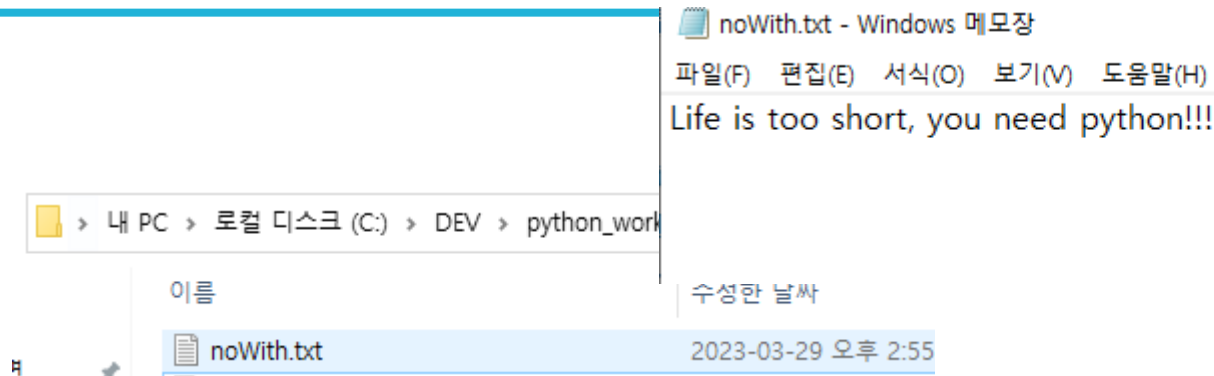
with문과 함께 사용하기

- 파일을 열면 위와 같이 항상 close해 주는 것이 좋다. 하지만 이렇게 파일을 열고 닫는 것을 자동으로 처리할 수 있다면 편리하지 않을까? 파이썬의 with문이 바로 이런 역할을 해준다.

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>F5=open("datasets/noWith.txt",'w') F5.write("Life is too short, you need python!!!") F5.close()</pre>

결과값1

비고





4. 파일 입/출력

with문과 함께 사용하기

- 파일을 열면 위와 같이 항상 close해 주는 것이 좋다. 하지만 이렇게 파일을 열고 닫는 것을 자동으로 처리할 수 있다면 편리하지 않을까? 파이썬의 with문이 바로 이런 역할을 해준다.

파일	소스코드
실습환경	py3_10_basic

소스코드	<pre>with open("datasets/With.txt",'w') as F6: F6.write("Life is too short, you need python!!!")</pre>
------	--

결과값1

내 PC > 로컬 디스크 (C:) > DEV > python_works > data

이름

수정한 날짜

With.txt

2023-03-29

With.txt - Windows 메모장

비고

파일(F) 편집(E) 서식(O) 보기(V) 도움말(H)

Life is too short, you need python!!!



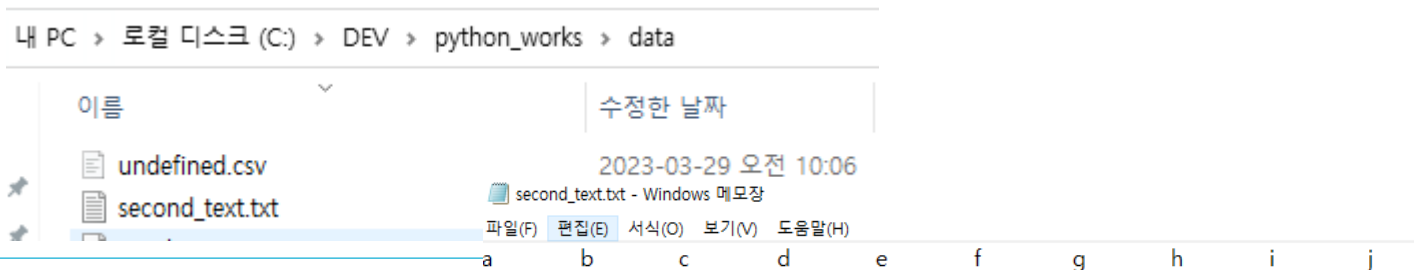
4. 파일 입/출력

텍스트 파일 쓰기

파일	소스코드
실습환경	py3_10_basic
소스코드	<pre>my_letters = ['a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'] max_index = len(my_letters) filewriter = open("datasets/second_text.txt", 'w') for index_value in range(len(my_letters)): if index_value < (max_index-1): filewriter.write(my_letters[index_value]+'t') else: filewriter.write(my_letters[index_value]+'n') filewriter.close() print("Output ###: Output written to file")</pre>

Output ###: Output written to file

결과값1



비고

Pickle 개요

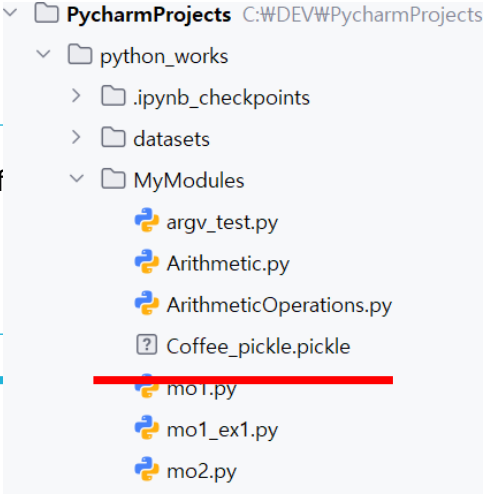
- 프로그램은 실행이 끝나버리면 모든 데이터가 사라진다고 했으므로 끝나기 전에 어딘가 저장을 해야한다. 이 때 사용할 수 있는 것이 바로 pickle . pickle 은 프로그램에서 사용하고 있는 데이터를 파일 형태로 저장하거나 불러올 수 있게 해주는 모듈이다.
- 먼저 pickle 을 이용하여 데이터를 파일로 저장을 할 때는 dump() 라는 함수를 사용. 첫 번째 전달 값으로는 저장할 데이터를, 두 번째 전달 값으로는 데이터를 저장할 파일을 적어준다.

`dump(data, dest_file)`

5. pickle

Pickle 개요

- 코드를 실행하면 profile 에 들어있는 데이터가 출력되고 워크스페이스 내에 Coffee_pickle.pickle 이라는 파일이 생긴다.

파일	소스코드	비고
실습환경	py3_10_basic > Coffee_pickle.py	
소스코드	<pre>import pickle # pickle 모듈 가져다 쓰기 profile_file = open("MyModules/Coffee_pickle.pickle", "wb") # 바이너리(binary) 형태로 저장 profile = {"Coffee Shop": "My Coffee Shop", "Menu": ["Coffee", "Tea", "Smoothie"]} print(profile) pickle.dump(profile, profile_file) # profile 데이터를 file 에 저장 profile_file.close()</pre>	
결과값1	<pre>(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> python .\Coffee_pickle.py {'Coffee Shop': 'My Coffee Shop', 'Menu': ['Coffee', 'Tea', 'Smoothie']}</pre> <pre>(py3_10_basic) PS C:\DEV\PycharmProjects\python_works></pre>	
비고		

5. pickle

Pickle 개요

- load() 함수를 이용하고 전달값으로는 파일을 작성하면 된다.
 - load(src_file)

파일	소스코드	비고
실습환경	py3_10_basic > Coffee_pickle.py import pickle # pickle 모듈 가져다 쓰기	
소스코드	profile_file = open("Mymodules/Coffee_pickle.pickle", "rb") # 읽을 때에도 바이너리(binary) 명시 profile = pickle.load(profile_file) # file 에 있는 정보를 불러와서 profile 에 저장 print(profile) profile_file.close()	
결과값1	<pre>import pickle # pickle 모듈 가져다 쓰기 profile_file = open("MyModules/Coffee_pickle.pickle", "rb") # 읽을 때에도 바이너리(binary) 명시 profile = pickle.load(profile_file) # file 에 있는 정보를 불러와서 profile 에 저장 print(profile) profile_file.close() {'Coffee Shop': 'My Coffee Shop', 'Menu': ['Coffee', 'Tea', 'Smoothie']}</pre>	
비고	코드를 실행해보면 저장할 때와 동일한 데이터를 그대로 불러온 것을 확인할 수 있다.	

문제

- **당신의 회사에서는 매주 1회 작성해야 하는 보고서가 있습니다.**
 - **보고서는 항상 아래와 같은 형태로 출력되어야 합니다.**
- X 주차 주간보고 -
 - 부서 :
 - 이름 :
 - 업무 요약 :
- **1주차부터 50주차까지의 보고서 파일을 만드는 프로그램을 작성하시오.**
- **조건 : 파일명은 '1주차.txt', '2주차.txt', ... 와 같이 만듭니다.**

답

파일	소스코드	비고
실습환경	py3_10_basic	
소스코드	<pre>for i in range(1, 51): with open(str(i) + "주차.txt", "w", encoding="utf8") as report_file: report_file.write("- {0} 주차 주간보고 -".format(i)) report_file.write("\n부서 : ") report_file.write("\n이름 : ") report_file.write("\n업무 요약 : ")</pre>	

내 PC > 로컬 디스크 (C:) > DEV > python_works >

결과값1

이름	수정한 날짜	유형	크기
1주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
2주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
3주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
4주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
5주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
6주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
7주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
8주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
9주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
10주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
11주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
12주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB
13주차.txt	2023-04-02 오후 12:03	텍스트 문서	1KB

비고

『2-3』

빅데이터 분석 도구 -advanced

- 파이썬 표준 입/출력
- 웹-Flask 개발 환경 만들기
- ...





1. Syllabus

학습목표

- 이 워크샵에서는 PyQt 에 대해 알 수 있습니다.
- 또한 웹-Flask 개발 환경 만들기에 대해 알 수 있습니다.

눈높이 체크

- 웹의 개념을 알고 계신가요?

PyQt 개요

- PyQt는 파이썬 플러그인으로 구현된 크로스 플랫폼 GUI 툴킷 Qt의 파이썬 바인딩이다. PyQt는 영국 회사 리버뱅크 컴퓨팅이 개발한 자유 소프트웨어이다. PyQt는 마이크로소프트 윈도우, 다양한 유닉스 파생판(리눅스, 맥OS 포함)을 지원한다

PyQt5 패키지 설치와 창 띄우기

- PyQt-1.py

```
import sys
from PyQt5.QtWidgets import *
```

```
app = QApplication(sys.argv)
qw = QWidget()
qw.show()
sys.exit(app.exec_())
```

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-1.py
```

```
Traceback (most recent call last):
```

```
File "PyQt-1.py", line 2, in <module>
```

```
    from PyQt5.QtWidgets import *
```

```
ModuleNotFoundError: No module named 'PyQt5'
```

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works>
```

PyQt5 패키지 설치와 창 띄우기

(py3_10_basic) C:\DEV\PycharmProjects\python_works> pip install pyqt5

Collecting pyqt5

Downloading PyQt5-5.15.9-cp37-abi3-win_amd64.whl (6.8 MB)

6.8/6.8 MB 6.3 MB/s eta 0:00:00

Collecting PyQt5-sip<13,>=12.11

Downloading PyQt5_sip-12.11.1-cp38-cp38-win_amd64.whl (78 kB)

78.1/78.1 kB 4.5 MB/s eta 0:00:00

Collecting PyQt5-Qt5>=5.15.2

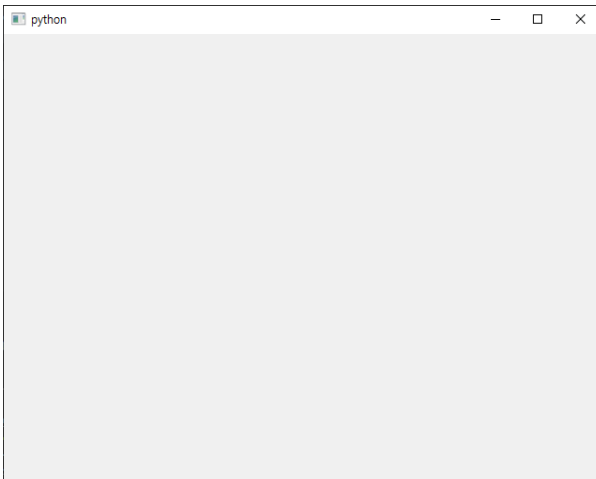
Downloading PyQt5_Qt5-5.15.2-py3-none-win_amd64.whl (50.1 MB)

50.1/50.1 MB 2.7 MB/s eta 0:00:00

Installing collected packages: PyQt5-Qt5, PyQt5-sip, pyqt5

Successfully installed PyQt5-Qt5-5.15.2 PyQt5-sip-12.11.1 pyqt5-5.15.9

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-1.py



간단한 예제

● PyQt-2.py

```
import sys
from PyQt5.QtWidgets import QApplication, QWidget, QPushButton

def main():
    app = QApplication(sys.argv)
    widgets = QWidget()
    widgets.resize(200,50)
    widgets.setWindowTitle("PyQt5 GUI")
    # 푸시 버튼
    btn1 = QPushButton('O', widgets)
    btn1.resize(100,30)
    btn1.move(0,20)
    btn2 = QPushButton('X', widgets)
    btn2.resize(100,30)
    btn2.move(100,20)
    # 푸시 버튼 클릭
    btn1.clicked.connect(lambda: print("O 클릭"))
    btn2.clicked.connect(lambda: print("X 클릭"))
    widgets.show()
    sys.exit(app.exec_())
main()
```

1. PyQt

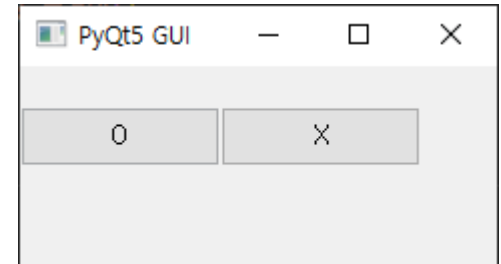
간단한 예제

- PyQt-2.py

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-2.py

O 클릭
X 클릭

(py3_10_basic) C:\DEV\PycharmProjects\python_works>



위젯의 타이틀 및 배경색 설정하기

● PyQt-3.py

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt3Widget import PyQt3Widget
app = QApplication(sys.argv)
myw = PyQt3Widget ()
myw.show()
sys.exit(app.exec_())
```

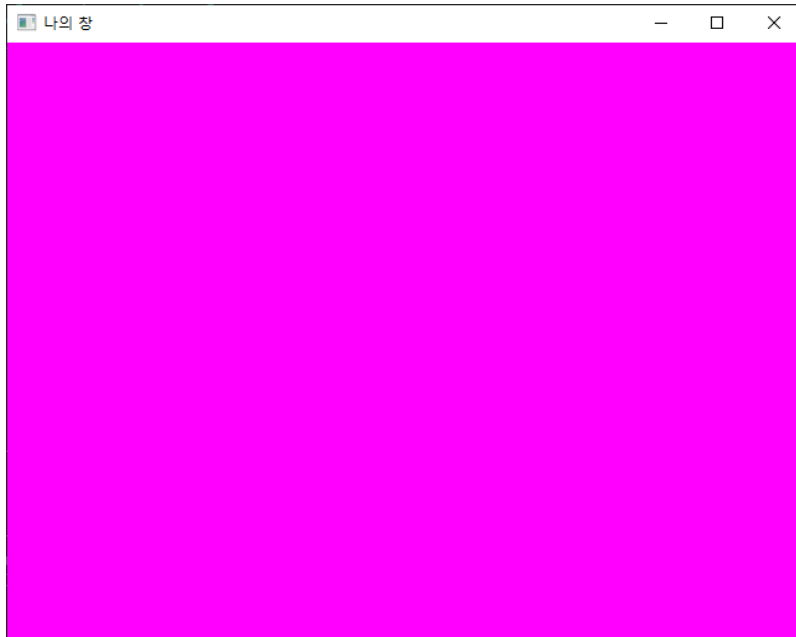
● PyQt3Widget.py

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt3Widget (QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("나의 창")
        pal = QPalette()
        pal.setColor(QPalette.Background,QColor(255,0,255))
        self.setAutoFillBackground(True)
        self.setPalette(pal)
```

1. PyQt

위젯의 타이틀 및 배경색 설정하기

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-3.py





1. PyQt

이벤트 핸들러

- PyQt-4.py

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt4Widget import PyQt4Widget
app = QApplication(sys.argv)
myw = PyQt4Widget ()
myw.show()
sys.exit(app.exec_())
```


이벤트 핸들러

● PyQt4Widget.py

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt4Widget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("테스트 입력 및 설정")
        self.te = QTextEdit(self)
        self.te.resize(400,90)
        self.resize(1000,600)
        self.btn = QPushButton("확인",self)
        self.btn.move(430,0)
        self.btn.resize(200,90)
        self.lb = QLabel("[테스트]",self)
        self.lb.move(0,110)
        self.btn.clicked.connect(self.BtnClick)
    def BtnClick(self):
        txt = self.te.toPlainText()
        self.lb.setText(txt)
        self.te.setText("")
```

1. PyQt

이벤트 핸들러

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-4.py



리스트 박스 사용하기

- PyQt-5.py

```
import sys
from PyQt5.QtWidgets import QApplication
from PyQt5.QtWidgets import QWidget
app = QApplication(sys.argv)
myw = QWidget ()
myw.show()
sys.exit(app.exec_())
```



1. PyQt

이벤트 핸들러

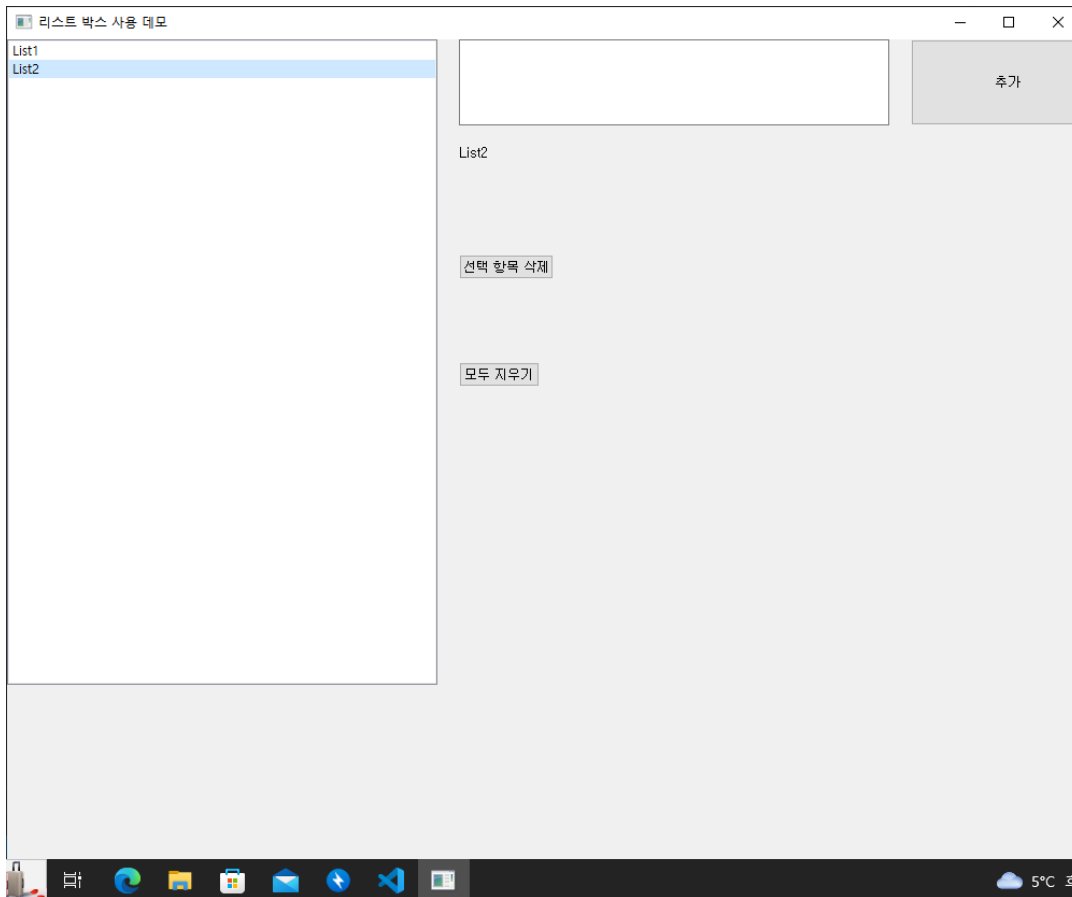
● PyQt5Widget.py

```
from PyQt5.QtWidgets import *
from PyQt5.QtGui import *
class PyQt5Widget(QWidget):
    def __init__(self):
        super().__init__()
        self.setWindowTitle("리스트 박스 사용 데모")
        self.resize(1000,800)
        self.lbox = QListWidget(self)
        self.lbox.resize(400,600)
        self.te=QTextEdit(self)
        self.te.move(420,0)
        self.te.resize(400,80)
        self.btn = QPushButton("추가",self)
        self.btn.move(840,0)
        self.btn.resize(180,80)
        self.btn.clicked.connect(self.AddItem)
        self.lb_sel = QLabel("[테스트]",self)
        self.lb_sel.move(420,100)
        self.lbox.currentItemChanged.connect(self.OnLBoxSelectChange)
        self.btn_remove = QPushButton("선택 항목 삭제",self)
        self.btn_remove.move(420,200)
        self.btn_remove.clicked.connect(self.RemoveItem)
        self.btn_clear = QPushButton("모두 지우기",self)
        self.btn_clear.move(420,300)
        self.btn_clear.clicked.connect(self.ClearAll)
    def ClearAll(self):
        self.lbox.clear()
    def RemoveItem(self):
        index = self.lbox.currentRow()
        self.lbox.takeItem(index)
    def AddItem(self):
        data = self.te.toPlainText()
        self.lbox.addItem(data)
        self.te.setText("")
    def OnLBoxSelectChange(self):
        item = self.lbox.currentItem()
        self.lb_sel.setText(item.text())
```

1. PyQt

이벤트 핸들러

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python PyQt-5.py





2. Tkinter

Tkinter 소개

- Tkinter는 Tcl/Tk에 대한 파이썬 Wrapper로서 Tcl/Tk를 파이썬에 사용할 수 있도록 한 Lightweight GUI 모듈이다. Tcl은 Tool Command Language의 약자로서 일종의 프로그래밍 언어이며, Tk는 크로스 플랫폼에 사용되는 일종의 GUI 툴킷이다. Tkinter는 타 GUI 프레임워크나 툴킷에 비해 지원되는 위젯들이 부족하고 UI도 그렇게 예쁘지 않다는 단점이 있지만, Python 설치시 기본적으로 내장되어 있는 파이썬 표준 라이브러리이기 때문에 쉽고 간단한 GUI 프로그램을 만들 때 활용될 수 있다.



2. Tkinter

Tkinter 소개

- 주의할 점은 라이브러리 이름이 tkinter라고 해서 pip install tkinter라고 입력하면 설치되지 않습니다.
- pip install tk라고 입력해야 합니다.

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works> pip install tkinter  
ERROR: Could not find a version that satisfies the requirement tkinter (from versions: none)  
ERROR: No matching distribution found for tkinter
```

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works> pip install tk  
Collecting tk  
  Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)  
Installing collected packages: tk  
Successfully installed tk-0.1.0
```

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works>
```



2. Tkinter

Tkinter 설치와 기본 문장

- Tkinter는 파이썬에 기본 내장되어 있기 때문에 PyQt처럼 별도로 설치할 필요가 없다. Tkinter를 사용하기 위해서는 먼저 tkinter 모듈을 아래와 같이 import 해야 한다 (주: Python 2에서는 Tkinter를 import 하고, Python 3에서는 tkinter를 import 한다). tkinter 모듈을 import한 다음에는 Tk 클래스 객체(root)를 생성하고, 이 객체의 mainloop() 메서드를 호출한다. 아래 코드와 같이 이런 기본 문장들을 수행하면, 빈 다이얼로그가 화면에 표시된다.

● Tk-1.py

```
from tkinter import *  
root = Tk()  
root.mainloop()
```

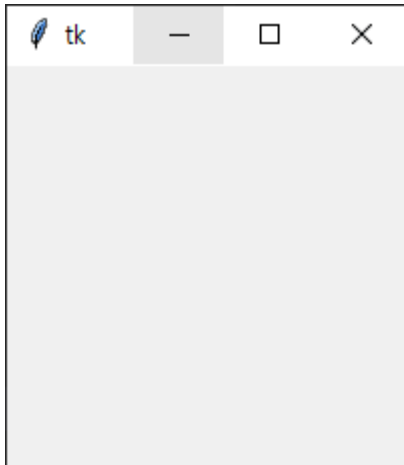



2. Tkinter

Tkinter 설치와 기본 문장

- `mainloop()`는 이벤트 메시지 루프로서 키보드나 마우스 혹은 화면 Redraw와 같은 다양한 이벤트로부터 오는 메시지를 받고 전달하는 역할을 한다.

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python Tk-1.py



2. Tkinter

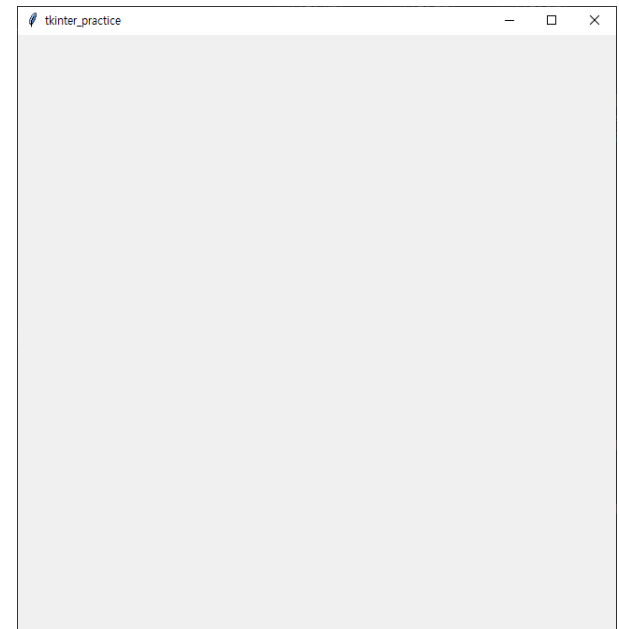
타이틀 지정 및 크기설정

- 타이틀은 위에서 본 윈도우 창의 제목을 말한다. 처음 아무 속성을 적지 않으면 tk로 지정되는데, 이것을 title 속성을 이용하여 바꿀 수 있다.
- 크기는 geometry 속성을 이용하여 바꿀 수 있다. 이때 파이썬의 *가 아닌 x로 사용해야 한다.
- 윈도우의 이름은 tkinter_practice로 바뀌었으며, 크기는 640x640으로 아주 커진 모습을 볼 수 있다.

● Tk-2-1.py

```
from tkinter import *
```

```
root = Tk() # GUI 생성  
root.title("tkinter_practice") #상단의 타이틀 지정  
root.geometry("640x640") # 크기 설정 (640x640)  
root.mainloop() # GUI가 보이고 종료될때까지 실행함
```





2. Tkinter

타이틀 지정 및 크기설정

- 문자열 출력

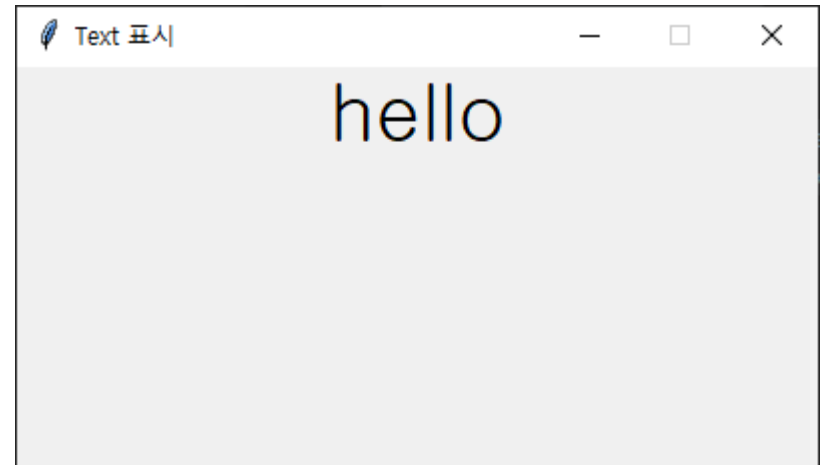
- Tk-2-2.py

```
import tkinter
import tkinter.font

root = tkinter.Tk()
root.title("Text 표시")
root.geometry("400x200")
root.resizable(False,False)

font = tkinter.font.Font(size = 30)
label=tkinter.Label(root, text="hello", font=font)
label.pack()

root.mainloop()
```



2. Tkinter

타이틀 지정 및 크기설정

● 타이머 출력

◦ Tk-2-3.py

```
import tkinter
import tkinter.font
```

```
root = tkinter.Tk()
root.title("타이머 출력")
root.geometry("400x200")
root.resizable(False,False)
```

```
font = tkinter.font.Font(size = 30)
label=tkinter.Label(root, text="", font=font)
label.pack()
```

```
cnt = 0
def get_coin_1sec():
    global cnt
    now_btc_price = str(cnt)
    cnt = cnt + 1
    label.config(text=now_btc_price)
    root.after(1000,get_coin_1sec)
```

```
get_coin_1sec()
```

```
root.mainloop()
```



2. Tkinter

버튼 생성

- Btn1은 (현재 윈도우 , 버튼 이름) 으로 설정하였다. 위에서 `root = Tk()`로 윈도우를 생성하였기 때문에 현재윈도우는 `root`이고 버튼 이름은 각자 설정하면 된다.
 - Btn2는 (현재윈도우 , 넓이 , 높이 , 버튼이름) 으로 설정하였다.

(py3_10_basic) C:\DEV\PycharmProjects\python_works> python Tk-3.py





2. Tkinter

버튼 생성

- Tk-3.py

```
from tkinter import *
```

```
root = Tk() # GUI 생성  
root.title("tkinter_practice") #상단의 타이틀 지정  
root.geometry("640x640") # 크기 설정 (640x640)
```

```
btn1 = Button(root, text = "기본버튼") #root로 지정한 윈도우에 button 생성  
btn1.pack() # 윈도우상에 상대 위치로 위젯을 배치
```

```
btn2 = Button(root, width = 10 , height = 10 , text="크기설정버튼")  
btn2.pack()
```

```
root.mainloop() # GUI가 보이고 종료될때까지 실행함
```

간단한 사칙 연산 계산기

파일	소스코드	비고
실습환경	<pre>py3_10_basic > calculator.py def add(x, y): return x + y def subtract(x, y): return x - y def multiply(x, y): return x * y def divide(x, y): return x / y print("사칙연산을 선택 하세요.") print("1.더하기") print("2.빼기") print("3.곱하기") print("4.나누기") choice = input("선택 하세요(1/2/3/4):") num1 = int(input("첫번째 숫자 : ")) num2 = int(input("두번째 숫자 : "))</pre>	

간단한 사칙 연산 계산기

파일	소스코드	비고
----	------	----

실습환경	py3_10_basic	
------	--------------	--

소스코드	<pre>if choice == '1': print(num1,"+",num2,"=", add(num1,num2)) elif choice == '2': print(num1,"-",num2,"=", subtract(num1,num2)) elif choice == '3': print(num1,"*",num2,"=", multiply(num1,num2)) elif choice == '4': print(num1,"/",num2,"=", divide(num1,num2)) else: print("잘못된 선택")</pre>	
------	--	--

결과값1		
------	--	--

	<pre>(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> python .\calculator.py 사칙연산을 선택 하세요. 1.더하기 2.빼기 3.곱하기 4.나누기 선택 하세요(1/2/3/4):1 첫번째 숫자 : 5 두번째 숫자 : 3 5 + 3 = 8 (py3_10_basic) PS C:\DEV\PycharmProjects\python_works></pre>	
--	--	--

비고		
----	--	--

Tkinter를 사용한 GUI 계산기

파일	소스코드	비고
----	------	----

실습환경		
------	--	--

	<pre>py3_10_basic > TkinterGUI.py import tkinter as tkt root = tkt.Tk() root.title("계산기") root.resizable(0, 0) root.wm_attributes("-topmost", 1) # 변수 선언 equa = "" equation = tkt.StringVar() calculation = tkt.Label(root, textvariable=equation) equation.set("계산식을 입력하세요 : ") calculation.grid(row=2, columnspan=8)</pre>	<pre># 윈도우 크기 고정한다 # 화면 상단에 유지된다.</pre>
--	--	---

소스코드	
------	--

	<pre>def btnPress(num): global equa equa = equa + str(num) equation.set(equa) def EqualPress(): global equa total = str(eval(equa)) equation.set(total) equa = "" def ClearPress(): global equa equa = "" equation.set("")</pre>
--	--

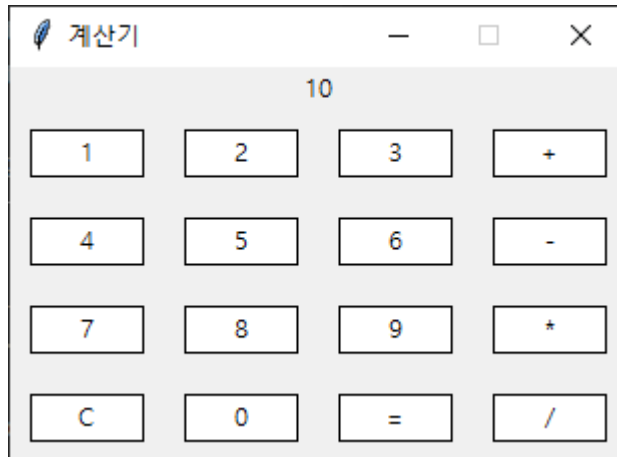
Tkinter를 사용한 GUI 계산기

파일	소스코드	비고
실습환경	py3_10_basic	
소스코드	<pre>Button0 = tkt.Button(root, text="0", bg='white',command=lambda: btnPress(0), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button0.grid(row=6, column=2, padx=10, pady=10) Button1 = tkt.Button(root, text="1", bg='white',command=lambda: btnPress(1), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button1.grid(row=3, column=1, padx=10, pady=10) Button2 = tkt.Button(root, text="2", bg='white',command=lambda: btnPress(2), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button2.grid(row=3, column=2, padx=10, pady=10) Button3 = tkt.Button(root, text="3", bg='white',command=lambda: btnPress(3), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button3.grid(row=3, column=3, padx=10, pady=10) Button4 = tkt.Button(root, text="4", bg='white',command=lambda: btnPress(4), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button4.grid(row=4, column=1, padx=10, pady=10) Button5 = tkt.Button(root, text="5", bg='white',command=lambda: btnPress(5), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button5.grid(row=4, column=2, padx=10, pady=10) Button6 = tkt.Button(root, text="6", bg='white',command=lambda: btnPress(6), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button6.grid(row=4, column=3, padx=10, pady=10) Button7 = tkt.Button(root, text="7", bg='white',command=lambda: btnPress(7), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button7.grid(row=5, column=1, padx=10, pady=10) Button8 = tkt.Button(root, text="8", bg='white',command=lambda: btnPress(8), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button8.grid(row=5, column=2, padx=10, pady=10) Button9 = tkt.Button(root, text="9", bg='white',command=lambda: btnPress(9), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Button9.grid(row=5, column=3, padx=10, pady=10) Plus = tkt.Button(root, text="+", bg='white',command=lambda: btnPress("+"), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Plus.grid(row=3, column=4, padx=10, pady=10) Minus = tkt.Button(root, text="-", bg='white',command=lambda: btnPress("-"), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Minus.grid(row=4, column=4, padx=10, pady=10) Multiply = tkt.Button(root, text="*", bg='white',command=lambda: btnPress("*"), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Multiply.grid(row=5, column=4, padx=10, pady=10) Divide = tkt.Button(root, text="/", bg='white',command=lambda: btnPress("/"), height=1, width=7, borderwidth=1, relief=tkt.SOLID) Divide.grid(row=6, column=4, padx=10, pady=10) Equal = tkt.Button(root, text "=", bg='white',command=EqualPress, height=1, width=7, borderwidth=1, relief=tkt.SOLID) Equal.grid(row=6, column=3, padx=10, pady=10) Clear = tkt.Button(root, text="C", bg='white',command=ClearPress, height=1, width=7, borderwidth=1, relief=tkt.SOLID) Clear.grid(row=6, column=1, padx=10, pady=10) root.mainloop()</pre>	
결과값1		
비고		

Tkinter를 사용한 GUI 계산기

파일	소스코드	비고
실습환경	py3_10_basic	
소스코드	(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> python .\TkinterGUI.py	

결과값1



비고

3. flask 가상환경

flask 패키지 설치

```
(py3_10_basic) C:\DEV\PycharmProjects\python_works> pip install flask
```

```
Collecting flask
```

```
  Downloading Flask-2.2.3-py3-none-any.whl (101 kB)
```

```
101.8/101.8 kB 1.5 MB/s eta 0:00:00
```

```
Collecting importlib-metadata>=3.6.0
```

```
...
```

```
Installing collected packages: zipp, typing-extensions, MarkupSafe, itsdangerous, colorama, Werkzeug, Jinja2, importlib-metadata, click, flask
Successfully installed Jinja2-3.1.2 MarkupSafe-2.1.2 Werkzeug-2.2.3 click-8.1.3 colorama-0.4.6 flask-2.2.3 importlib-metadata-6.0.0 itsdangerous-2.1.2 typing-extensions-4.5.0
zipp-3.13.0
```

```
(py3_10_basic) PS C:\DEV\PycharmProjects\python_works> cd ..
```


```
(py3_10_basic) PS C:\DEV\PycharmProjects> mkdir flask_works
```

```
디렉터리: C:\DEV\PycharmProjects
```

Mode	LastWriteTime	Length	Name
d-----	2025-05-10 오후 1:47		flask_works

```
(py3_10_basic) PS C:\DEV\PycharmProjects> cd .\flask_works\
```

```
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works>
```



3. flask 가상환경

flask 패키지 설치 테스트

- hello1.py 작성

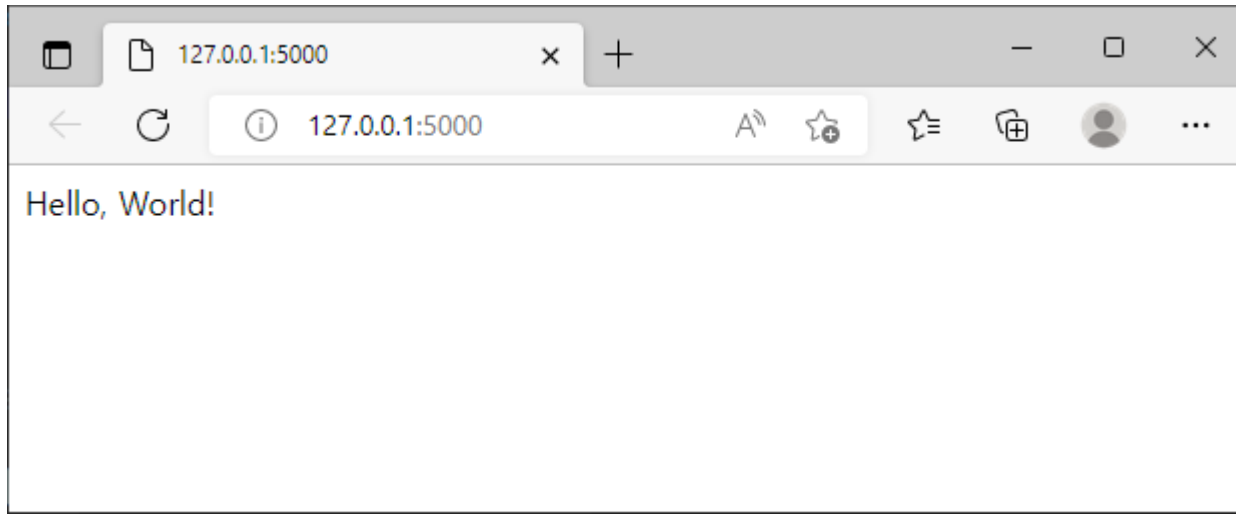
```
from flask import Flask
app = Flask(__name__)
@app.route("/")
def hello_world():
    return "<p>Hello, World!</p>"
```

3. flask 가상환경

flask 패키지 설치 테스트

- 터미널에서 다음 명령을 실행

```
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> $env:FLASK_APP = "hello1.py"
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> flask run
* Serving Flask app 'hello1.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [10/May/2025 13:55:17] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2025 13:55:17] "GET /favicon.ico HTTP/1.1" 404 -
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> ^C
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works>
```





3. flask 가상환경

hello1.py를 다음처럼 쓸 수 있다

- hello2.py 작성

```
from flask import Flask
app = Flask(__name__) # __main__

@app.route('/') # 접속할 주소
def index():
    return 'Hello~~~!!!' # 브라우저에 보여지는 응답결과

if __name__ == '__main__':
    app.run(debug=True) # 서버 자동 재실행 debug=True 옵션
```

3. flask 가상환경

hello1.py를 다음처럼 쓸 수 있다

```
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> $env:FLASK_APP = "hello2.py"
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> flask run
* Serving Flask app 'hello2.py'
* Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
127.0.0.1 - - [10/May/2025 13:58:40] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/May/2025 13:58:40] "GET /favicon.ico HTTP/1.1" 404 -
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> ^C
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works> ^C
(py3_10_basic) PS C:\DEV\PycharmProjects\flask_works>
```



『2-4』 빅데이터 분석 도구 -Self 점검

- 필기 평가
- 실기 평가





1. Quiz1

대화하는 프로그램 만들기

자동 판매기를 시뮬레이션하는 프로그램을 작성하여 보자. 사용자는 1000원짜리 지폐와 500원짜리 동전, 100원짜리 동전을 사용할 수 있다. 물건값을 입력하고 1000원권, 500원짜리 동전, 100원짜리 동전의 개수를 입력하면 거스름돈을 계산하여서 동전으로 반환한다.

물건값을 입력하시오: 750

1000원 지폐개수: 1

500원 동전개수: 0

100원 동전개수: 0

500원 = 0 100원 = 2 10원 = 5 1원 = 0



1. Quiz1

답

```
itemPrice = int(input("물건값을 입력하시오: "))
note = int(input("1000원 지폐개수: "))
coin500 = int(input("500원 동전개수: "))
coin100 = int(input("100원 동전개수: "))

change = note*1000 + coin500*500 + coin100*100 - itemPrice

# 거스름돈(500원 동전 개수)을 계산한다.
nCoin500 = change//500
change = change%500

# 거스름돈(100원 동전 개수)을 계산한다.
nCoin100 = change//100
change = change%100

# 거스름돈(10원 동전 개수)을 계산한다.
nCoin10 = change//10
change = change%10

# 거스름돈(1원 동전 개수)을 계산한다.
nCoin1 = change

print("500원=", nCoin500, "100원=", nCoin100, "10원=", nCoin10, "1원=", nCoin1)
```

```
물건값을 입력하시오: 10
1000원 지폐개수: 10
500원 동전개수: 50
100원 동전개수: 10
500원= 71 100원= 4 10원= 9 1원= 0
```



2. Quiz2

연락처 관리 프로그램

파이썬을 이용하여 연락처를 관리하는 프로그램을 작성하여 보자. 연락처 관리 프로그램은 다음과 같은 메뉴를 가져야 한다.

```
-----
1. 친구 리스트 출력
2. 친구추가
3. 친구삭제
4. 이름변경
9. 종료
메뉴를 선택하시오: 2
이름을 입력하시오: 홍길동
-----
1. 친구 리스트 출력
2. 친구추가
3. 친구삭제
4. 이름변경
9. 종료
메뉴를 선택하시오: 1
['홍길동']
-----
...
```



2. Quiz2

답

```
menu = 0
friends = []
while menu != 9:
    print("-----")
    print("1. 친구 리스트 출력")
    print("2. 친구추가")
    print("3. 친구삭제")
    print("4. 이름변경")
    print("9. 종료")
    menu = int(input("메뉴를 선택하시오: "))
    if menu == 1:
        print(friends)
    elif menu == 2:
        name = input("이름을 입력하시오: ")
        friends.append(name)
menu = 0
friends = []
while menu != 9:
    print("-----")
    print("1. 친구 리스트 출력")
    print("2. 친구추가")
    print("3. 친구삭제")
    print("4. 이름변경")
    print("9. 종료")
    menu = int(input("메뉴를 선택하시오: "))
    if menu == 1:
        print(friends)
    elif menu == 2:
        name = input("이름을 입력하시오: ")
        friends.append(name)
```

```
-----
1. 친구 리스트 출력
2. 친구추가
3. 친구삭제
4. 이름변경
9. 종료
메뉴를 선택하시오: 1
```



3. Quiz3

파일에서 중복되지 않은 단어의 개수

텍스트 파일을 읽어서 단어를 얼마나 다양하게 사용하여 문서를 작성하였는지를 계산하는 프로그램을 작성해보자.

입력 파일 이름: proverbs.txt

사용된 단어의 개수 = 18

{'travels', 'half', 'that', 'news', 'alls', 'well', 'fast',
'feather', 'flock', 'bad', 'together', 'ends', 'is', 'a', 'done',
'begun', 'birds', 'of'}



3. Quiz3

답

```
# 단어에서 구두점을 제거하고 소문자로 만든다.
def process(w):
    output = ""
    for ch in w:
        if( ch.isalpha() ):
            output += ch
    return output.lower()

words = set()
# 파일을 연다.
fname = input("입력 파일 이름: ")
file = open(fname, "r")
# 파일의 모든 줄에 대하여 반복한다.
for line in file:
    lineWords = line.split()
    for word in lineWords:
        words.add(process(word)) # 단어를 세트에 추가한다.
print("사용된 단어의 개수=", len(words))
print(words)
```


4. Quiz4

단어 카운터

사용자가 지정하는 파일을 읽어서 파일에 저장된 각각의 단어가 몇 번이나 나오는지 계산하는 프로그램을 작성하여 보자.

파일 이름: proverbs.txt

```
{'a': 1, 'done.': 1, 'that': 1, 'well.': 1, 'ends': 1, 'Well': 1,
'flock': 1, 'feather': 1, "All's": 1, 'Birds': 1, 'together.': 1,
'of': 1, 'fast.': 1, 'begun': 1, 'half': 1, 'well': 1, 'travels':
1, 'news': 1, 'is': 1, 'Bad': 1}
```

4. Quiz4

답

```
fname = input("파일 이름: ")
file = open(fname, "r")

table = dict()
for line in file:
    words = line.split()
    for word in words:
        if word not in table:
            table[word] = 1
        else:
            table[word] += 1

print(table)
```



5. Quiz5

은행 계좌

우리는 은행 계좌에 돈을 저금할 수 있고 인출할 수도 있다. 은행 계좌를 클래스로 모델링하여 보자. 은행 계좌는 현재 잔액 (balance)만을 인스턴스 변수로 가진다. 생성자와 인출 메소드 `withdraw()`와 저축 메소드 `deposit()` 만을 가정하자.

통장에서 100 가 출금되었음
통장에 10 가 입금되었음

5. Quiz5

답

```
class BankAccount:
    def __init__(self):
        self.__balance = 0

    def withdraw(self, amount):
        self.__balance -= amount
        print("통장에 ", amount, "가 입금되었음")
        return self.__balance

    def deposit(self, amount):
        self.__balance += amount
        print("통장에서 ", amount, "가 출금되었음")
        return self.__balance

a = BankAccount()
a.deposit(100)
a.withdraw(10)
```

통장에서 100 가 출금되었음
통장에 10 가 입금되었음
90

THANK YOU.

