

파이썬, 과연 엑셀보다 어려운가?

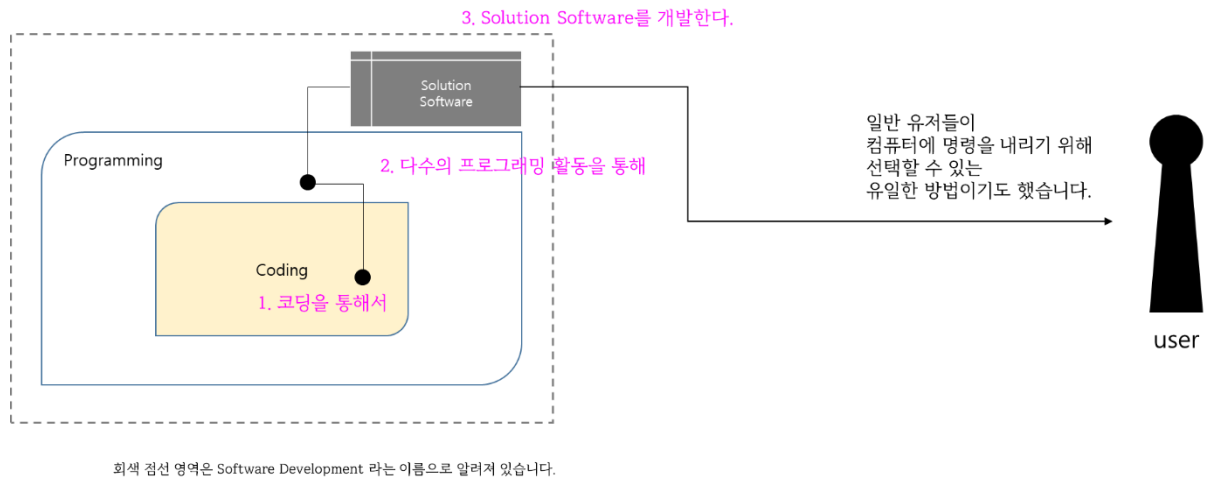
(플랜트 건축 설계팀-심화세미나 파이썬 소개 자료)

목차

| | |
|------------------------------|----|
| 도입: 들어가기 앞서..... | 1 |
| 엑셀 - 파이썬 함수 비교..... | 4 |
| 0. 주피터 노트북 열기 (인터넷 창으로)..... | 4 |
| 1. 사칙 연산..... | 7 |
| 2. Sum함수..... | 8 |
| 3. If 함수..... | 11 |
| 4. Vlookup 함수..... | 14 |
| 5. Left 함수..... | 19 |
| 결론..... | 21 |

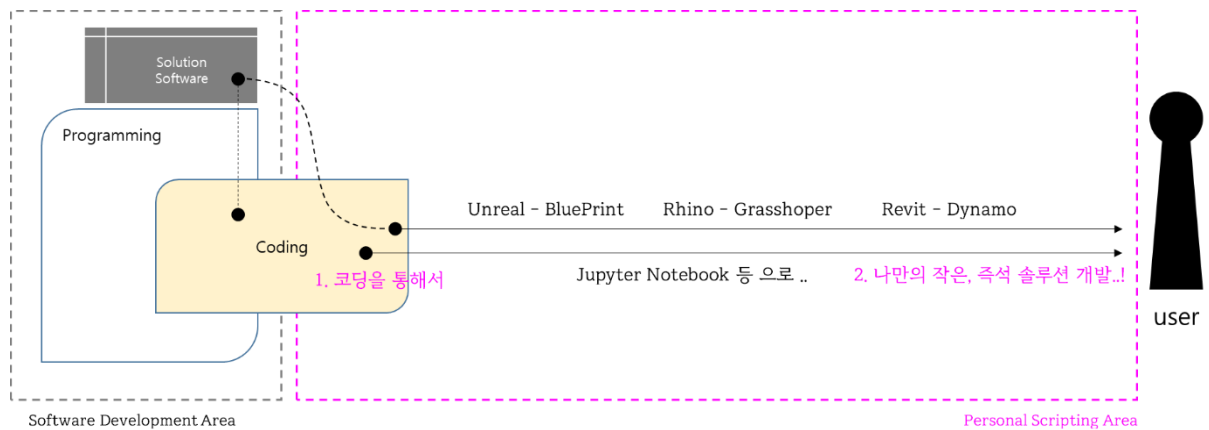
도입: 들어가기 앞서

누군가 자신의 문제를 컴퓨터로 해결하고자 한다면, 그 문제 해결의 흐름을 도식화했을 때 과거에는 다음과 같았을 겁니다.



쉽게 말하면 문제를 가지고 있는 user가 직접 컴퓨터에 명령 내릴 수단을 가지고 있지 않으니, 컴퓨터 코딩에 특화된 사람들이 1, 2, 3 단계를 수행해서 Solution Software를 개발한 뒤에 user에게 배포하는 식이었죠.

그런데 요즘은 다른 방식도 추가되었습니다.



위 그림처럼 기존의 소프트웨어 개발 영역도 여전히 존재하지만, 코딩을 통해 자신만의 즉석 솔루션을 개발해서 업무나 연구, 작업 등에 자동화, 효율화를 추구하게 되는 방향도 생겨나게 된 것이죠.

이걸 저 혼자 **Personal Scripting Area** 라고 부르는데, 이 영역에 속하는 툴들이 참 많이 있습니다.

그 종류 중 하나는 기존의 복잡하고 덩치가 큰 Solution Software 위에 올라타서, Solution Software 자체를 좀 더 효율적으로 조작할 수 있도록 도와주는 툴들이 있는가 하면 (Revit-Dynamo 같이),

Jupyter Notebook 같이 파이썬이나 여타 프로그래밍 언어를 사용자가 즉석에서 *REPL 환경으로 돌릴 수 있도록 만든 툴도 있습니다.

*REPL : Read-Eval-Print Loop 의 약자로 파이썬 대화형 환경 같은 것을 말합니다. 컴파일 과정 없이 즉석에서 코드를 입력하고 결과를 바로 알 수 있는 환경을 말합니다.

그러다 보니, 프로그래밍 언어를 배울 때 가장 장벽이 되던 개발환경 세팅에 대한 문제가 상당부분 사라지게 되었고, 사람들에게 Python 이 좀 더 가벼운 툴로 인식될 수 있는 환경이 갖춰진 것 같습니다.

이런 배경에서 생각해 보면,

순수한 언어 측면에서 봤을 때, 과연 파이썬 언어가 엑셀의 함수들을 배우는 것보다 고차원적인 두뇌 노동을 필요로 할까요?

저는 당연히 아니라고 생각합니다. 이제 파이썬에 대한 심리적 장벽을 좀 낮춰보자는 취지로 아래 글을 써 보도록 하겠습니다.

엑셀 - 파이썬 함수 비교

0. 주피터 노트북 열기 (인터넷 창으로)

일단 손쉽게 파이썬 코드를 작성해볼 수 있어야겠죠?

<https://jupyter.org/try> 를 눌러서 이동하세요.

Try Jupyter




Use our tools without installing anything

Project Jupyter builds tools, standards, and services for many different use cases. This page has links to interactive demos that allow you to try some of our tools for free online, thanks to [mybinder.org](#), a free public service provided by the Jupyter community.

Applications

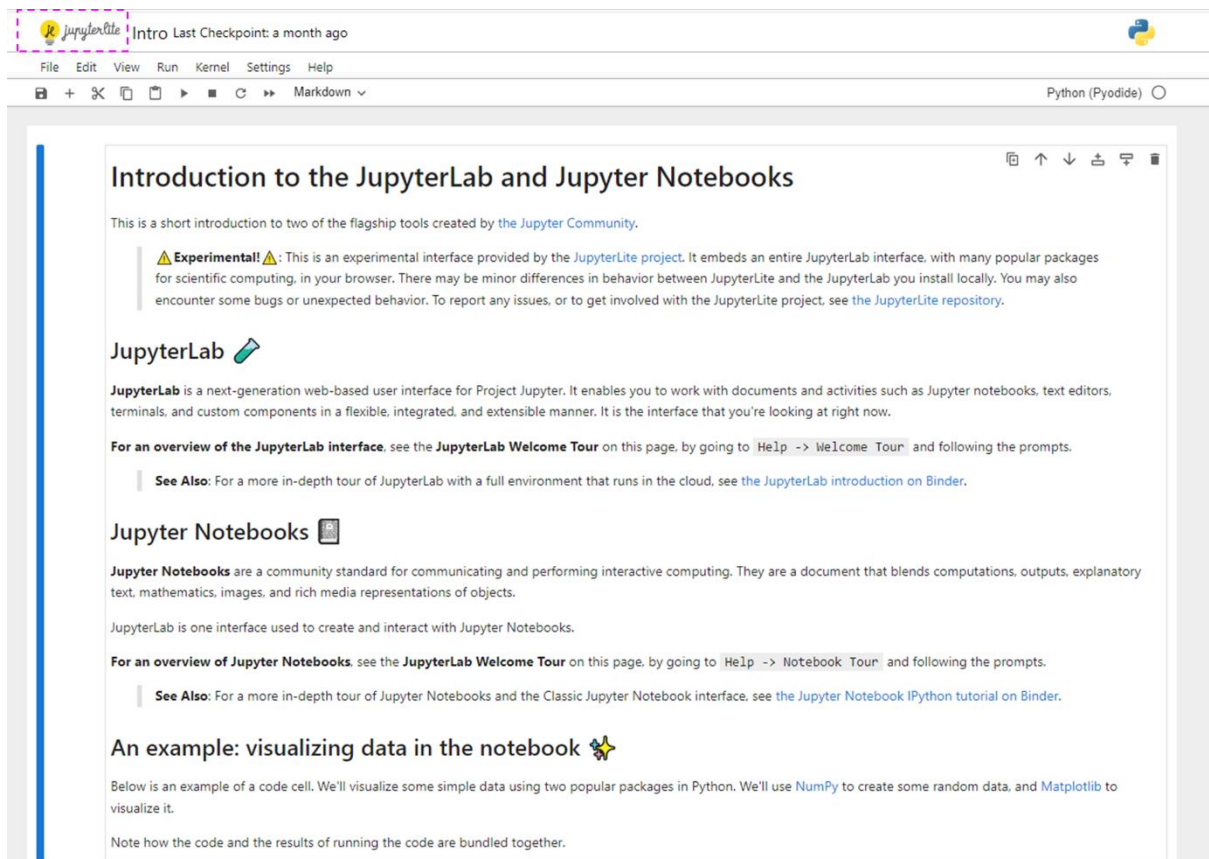
The Jupyter team builds several end-user applications that facilitate interactive computing workflows. Click the boxes below to learn how they work and to learn more. If you like one, you can find [installation instructions here](#).

⚠ Experimental ⚠ several of the environments below use the [JupyterLite project](#) to provide a self-contained Jupyter environment that runs in your browser. This is experimental technology and may have some bugs, so please be patient and report any unexpected behavior in [the JupyterLite repository](#).

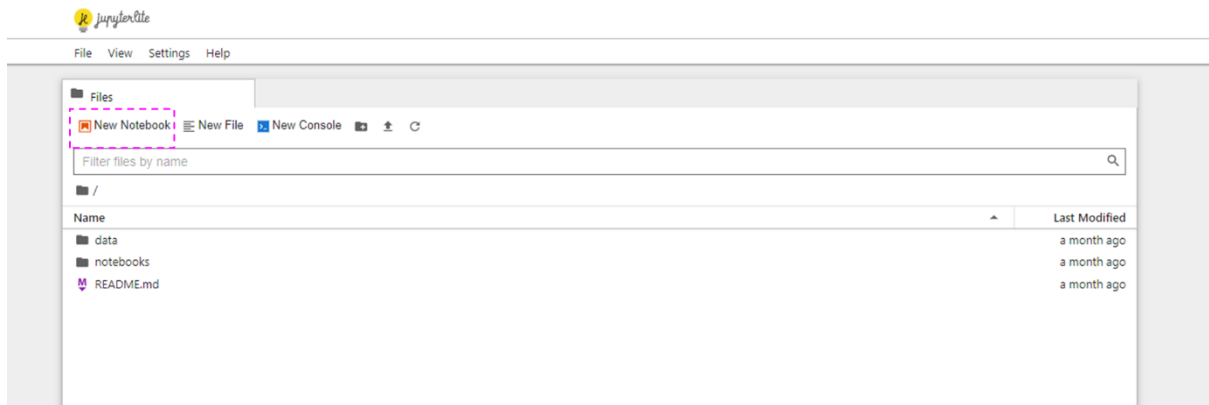
| JupyterLab | Jupyter Notebook | Voilà |
|---|---|---|
|  |  |  |
| The latest web-based interactive development environment | The original web application for creating and sharing computational documents | Share insights by converting notebooks into interactive dashboards |

3 개의 메뉴 중에 가운데 Jupyter Notebook 을 눌러보면 튜토리얼 주피터 노트북이 하나 뜹니다.

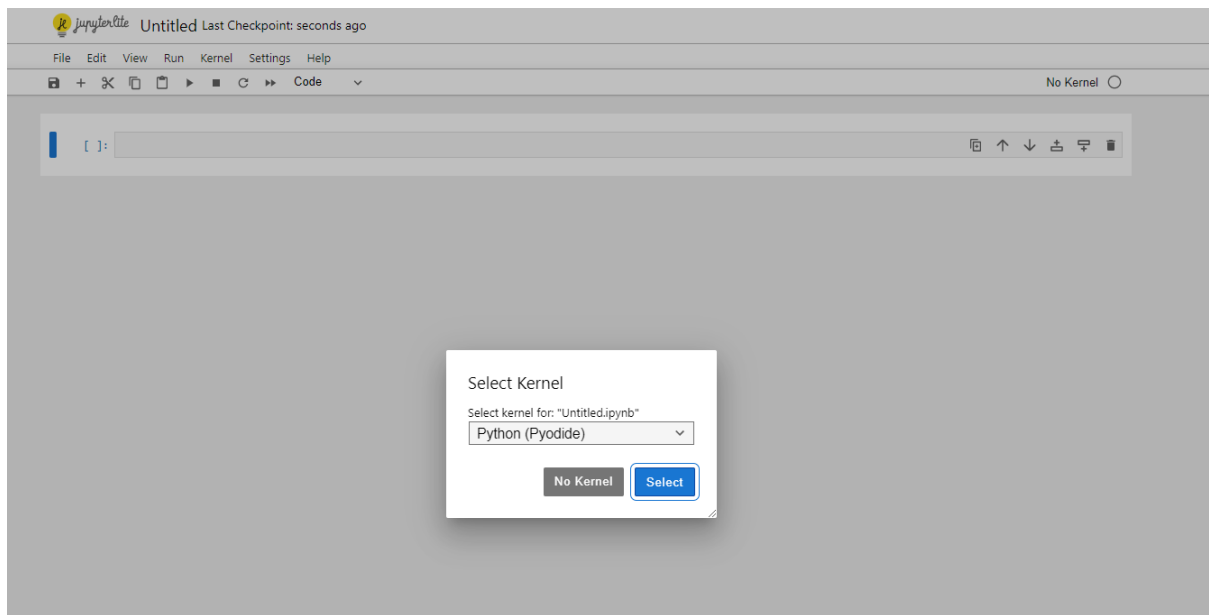
이건 나중에 한번 차분하게 읽어 보시고,



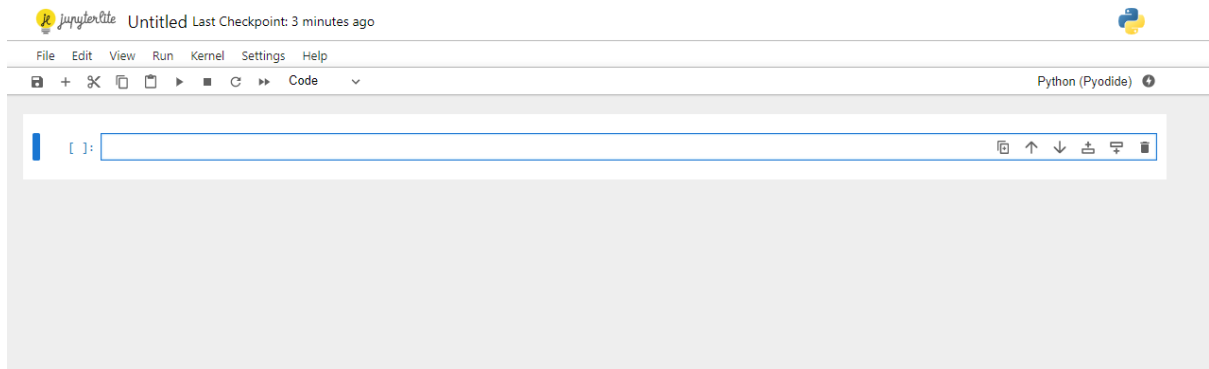
좌 상단에 있는 JupyterLite 로고를 눌러보세요.



이렇게 디렉토리가 나오고, 상단에 여러 메뉴가 있습니다. New Notebook 버튼을 누르면.



이렇게 빈 노트북이 나옵니다. 그리고 Select Kernel 창이 하나 뜨는데, 여기서 Python 이라고 되어 있는지 만 확인하고 Select 버튼을 눌러주면 됩니다.



자, 깔끔하게 빈 노트북 하나를 생성했습니다. 여기까지 1 분도 안 걸리셨을 거라 믿습니다.

만약 회사나 기타 네트워크 보안환경 등의 이유로 온라인 버전의 주피터노트북이 잘 작동되지 않는 경우에는 직접 컴퓨터에 주피터 노트북을 설치한 뒤에 실습해 보시기를 권합니다.

(어려우시면 유씨로 장만규 매니저에게 문의해 주세요.)

1. 사칙 연산

이제 엑셀도 새 창 하나 열어준 다음, 더하기, 빼기, 곱하기, 나누기부터 한번 보겠습니다.

| | | | | |
|----|---|---|---|---|
| C8 | | | | |
| | A | B | C | D |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | 3 | |
| 9 | | | | |
| 10 | | | | |

머리 속 떠오르는 데로 1 + 2 라고 적었는데, 3 으로 계산 결과가 잘 나왔습니다.

파이썬도 똑같습니다.

```
[1]: 1+2
[1]: 3
```

위쪽에 보이는 회색 칸 하나가 엑셀의 셀 한 칸에 해당한다고 생각하면 됩니다.

셀에 1+2 를 입력하고, 값 계산을 원한다는 의미로 shift + enter 키를 누르면 바로 아래에 결과값이 나타나게 됩니다.

그 밖에 빼기, 곱하기, 나누기는 생략하겠습니다.

2. Sum 함수

엑셀 Sum 함수도 해볼까요?

| | A | B | C | D | E |
|----|---|---|----|---|---|
| 1 | | | | | |
| 2 | | | | | |
| 3 | | | | | |
| 4 | | | | | |
| 5 | | | | | |
| 6 | | | | | |
| 7 | | | | | |
| 8 | | 4 | 28 | | |
| 9 | | 5 | | | |
| 10 | | 6 | | | |
| 11 | | 7 | | | |
| 12 | | 1 | | | |
| 13 | | 2 | | | |
| 14 | | 3 | | | |
| 15 | | | | | |

B 칼럼에 4,5,6,7,1,2,3 의 숫자들이 있는데, 한꺼번에 더하고 싶다면 Sum 함수를 쓰면 됩니다.

파이썬에서는?

```
[3]: data = [4,5,6,7,1,2,3]
```

```
[4]: sum(data)
```

```
[4]: 28
```

data 라는 변수에 [4, 5, 6, 7, 1, 2, 3] 이라는 숫자들을 리스트로 저장합니다. 그다음에 엑셀과 똑같이 sum() 이라는 함수이름 옆의 괄호에 data 라고 적어주면 결과값이 나옵니다. 별 차이가 없습니다.

data 라는 변수에 숫자들을 할당한 이유는 단순히 깔끔하게 보여주기 위해서입니다. sum([4, 5, 6, 7, 1, 2, 3]) 이라고 작성해도 동일하게 계산 결과가 나옵니다.

```
sum( [4, 5, 6, 7, 1, 2, 3] )
```

28

이렇게 말입니다.

단지 엑셀은 셀 하나에 숫자 하나씩 만 입력할 수 있기 때문에, 각 셀에 숫자를 집어넣고, 그 셀들의 위치 범위를 (B8:B14) 이렇게 입력하게 되는 것이 차이점입니다.

이제 sum(B8:B14) 과 sum(data)가 완전히 동일한 표현인 걸 이해할 수 있습니다.

Excel

=SUM (B8:B14)

=규칙 (재료)

Python

sum (data)

규칙 (재료)

컴퓨터에게 일을 시키기 위해 필요한 형식은 조금은 다를 수 있어도, 크게는 같습니다.

규칙 & 재료

재료에 규칙을 적용해서 바뀐 결과값을 받는다.

규칙과 재료, 다른 말로 하면 함수와 인자를 통해 컴퓨터에게 명령을 내릴 수 있습니다.

3. If 함수

엑셀에서 if 함수는 조건에 따라 다른 값을 반환하는 함수입니다.

=IF(조건, True일 때 반환할 값, False일 때 반환할 값)

이렇게 써 놓고 보니까 의외로 구조가 간단하죠?

If 함수에는 3 개의 재료만 입력해주면 됩니다.

- 조건
- 조건이 참일때 결과값
- 조건이 거짓일때의 결과값

조건이라는 말이 좀 애매하게 들릴 수 있는데, 계산해서 참인지 거짓인지 판별이 되는 구문이 조건이라고 생각하면 됩니다. 주로 등호 부등호를 통해 값을 비교하면 참이나 거짓이 값으로 나오겠죠?

| C8 | | | | | | | | |
|---|---|---|-------------|---|---|---|---|---|
| =IF(A8>B8, "왼쪽 숫자가 크다!", "오른쪽 숫자가 크다!") | | | | | | | | |
| | A | B | C | D | E | F | G | H |
| 1 | | | | | | | | |
| 2 | | | | | | | | |
| 3 | | | | | | | | |
| 4 | | | | | | | | |
| 5 | | | | | | | | |
| 6 | | | | | | | | |
| 7 | | | | | | | | |
| 8 | 7 | 4 | 왼쪽 숫자가 크다! | | | | | |
| 9 | 3 | 8 | 오른쪽 숫자가 크다! | | | | | |
| 10 | | | | | | | | |

간단한 예를 보여드리면 이렇게 됩니다.

이걸 파이썬으로 바꿔볼까요?

```
[6]: left = 7  
     right = 4
```

```
[9]: if left > right:  
     print("왼쪽 숫자가 크다!")  
     else:  
         print("오른쪽 숫자가 크다!")
```

왼쪽 숫자가 크다!

```
[10]: left = 3  
      right = 8
```

```
[11]: if left > right:  
     print("왼쪽 숫자가 크다!")  
     else:  
         print("오른쪽 숫자가 크다!")
```

오른쪽 숫자가 크다!

구조가 크게 다르지 않습니다. 조건을 if 라는 말 다음에 적어주고 콜론을 마지막에 붙인 뒤에, 줄을 바꿔 참일 때 결과를 적어줍니다.

다시 줄을 바꿔서 조건이 거짓일때 결과값은 else: 라고 적은 뒤에 또 줄 바꾸고 적어주면 됩니다.

if 조건:

True 일 때 반환할 값

else:

False 일 때 반환할 값

Python에서의 if 구문의 구조를 간단하게 써보면 위와 같습니다.

4. Vlookup 함수

엑셀에서 vlookup 함수는 특정 값을 참조하여 데이터 영역에서 해당 값과 일치하는 값을 찾아 반환하는 함수입니다.

=**VLOOKUP**(참조할 값, 참조할 범위, 반환할 열 번호, 정확히 일치 여부)

구조를 보면 vlookup 함수에는 4 개의 재료가 필요합니다.

- 참조할 값
- 참조할 범위
- 반환할 열 번호
- 정확히 일치 여부

4 번 항목은 거의 대부분 False 로 입력하면 됩니다. True로 하게 되면 유사한 값까지 포함해서 결과를 내기 때문에 원하는 것과 다른 결과가 나오는 경우가 생깁니다.

| D8 | | | | =VLOOKUP(C8,A8:B15,2,FALSE) | | | |
|----|----|------|----|-----------------------------|---|---|---|
| | A | B | C | D | E | F | G |
| 1 | | | | | | | |
| 2 | | | | | | | |
| 3 | | | | | | | |
| 4 | | | | | | | |
| 5 | | | | | | | |
| 6 | | | | | | | |
| 7 | | | | | | | |
| 8 | 김치 | 5000 | 양파 | 4000 | | | |
| 9 | 두부 | 2000 | | | | | |
| 10 | 만두 | 6000 | | | | | |
| 11 | 배추 | 3500 | | | | | |
| 12 | 양파 | 4000 | | | | | |
| 13 | 감자 | 3000 | | | | | |
| 14 | 수박 | 8000 | | | | | |
| 15 | 체리 | 8500 | | | | | |
| 16 | | | | | | | |

예제를 만들어 보면 이렇게 되겠네요. 노란색 셀에 양파라고 입력하면, 좌측 A, B 칼럼에 있는 데이터 중 첫째열을 기준으로 이름이 동일한 항목을 찾아서 B 열에 있는 가격을 알려줍니다.

이번에는 파이썬으로 동일한 작업을 해봐야겠죠?

딕셔너리를 쓰면 간단하게 구현할 수 있습니다.

먼저 엑셀과 동일하게 데이터 세팅을 해둘게요.

```
[17]: 이름 = [  
      "김치",  
      "두부",  
      "만두",  
      "배추",  
      "양파",  
      "감자",  
      "수박",  
      "체리"]
```

```
가격 = [  
      5000,  
      2000,  
      6000,  
      3500,  
      4000,  
      3000,  
      8000,  
      8500]
```

```
[18]: 이름
```

```
[18]: ['김치', '두부', '만두', '배추', '양파', '감자', '수박', '체리']
```

```
[19]: 가격
```

```
[19]: [5000, 2000, 6000, 3500, 4000, 3000, 8000, 8500]
```

이름 항목들과 가격 항목들이 파이썬에서 리스트로 저장되었습니다. 순서대로 대응되는 것을 확인할 수 있네요. 작업하기 편하게 아예 짝 지은 데이터 형식으로 바꿔 보겠습니다. 코드 한 줄이면 됩니다.


```
[20]: 참조값 = dict(zip(이름, 가격))
```

```
[21]: 참조값
```

```
[21]: {'김치': 5000,  
      '두부': 2000,  
      '만두': 6000,  
      '배추': 3500,  
      '양파': 4000,  
      '감자': 3000,  
      '수박': 8000,  
      '체리': 8500}
```

이렇게 짝지어진 데이터를 딕셔너리 라고 합니다.

엑셀 vlookup 함수에서는 이 작업을 하기 위해 참조할 범위, 반환할 열 번호 부분을 함수의 재료로 입력하게 되어 있다고 부연 설명 드릴 수 있겠네요.

이제 양파 가격을 출력해 볼까요?

```
[21]: 참조값
```

```
[21]: {'김치': 5000,  
      '두부': 2000,  
      '만두': 6000,  
      '배추': 3500,  
      '양파': 4000,  
      '감자': 3000,  
      '수박': 8000,  
      '체리': 8500}
```

```
[22]: 참조값["양파"]
```

```
[22]: 4000
```

```
[23]: 참조값["만두"]
```

```
[23]: 6000
```

```
[24]: 참조값["수박"]
```

```
[24]: 8000
```

딕셔너리의 이름 뒤에 대괄호를 배치하고, 그 안쪽에 재료의 이름을 넣기만 하면
알아서 해당하는 값을 출력해 줍니다.

```
[26]: 이름 = [
        "김치",
        "두부",
        "만두",
        "배추",
        "양파",
        "감자",
        "수박",
        "체리"]

    가격 = [
        5000,
        2000,
        6000,
        3500,
        4000,
        3000,
        8000,
        8500]

    참조값 = dict(zip(이름, 가격))
```

```
[27]: 참조값["양파"]
```

```
[27]: 4000
```

필요한 코드만 정리해서 다시 보여드리면 위 그림과 같습니다.

5. Left 함수

엑셀에서는 글자들 중의 일부를 가지고 조작하는 경우도 자주 있습니다. 대표적으로 Left, Right 함수들이 있습니다.

엑셀의 Left 함수는 문자열에서 왼쪽에서부터 지정한 길이만큼의 문자열을 반환하는 함수입니다.

| | | | | |
|----|---|--------|----|---|
| C8 | | | | |
| | A | B | C | D |
| 1 | | | | |
| 2 | | | | |
| 3 | | | | |
| 4 | | | | |
| 5 | | | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | 안녕하세요. | 안녕 | |
| 9 | | | | |

이런 식으로 사용합니다.

Python에서는 이 작업을 문자열 인덱싱이라는 기능으로 간단하게 구현할 수 있습니다.

```
[28]: 문자열 = "안녕하세요."
```

```
[31]: 문자열[0:2]
```

```
[31]: '안녕'
```

엄청 간단하죠? 구문을 해석해보면,

{ 문자열 이라는 변수에 “안녕하세요.” 를 저장한 뒤에,

문자열의 가장 0 번째 (프로그래밍에서는 가장 처음이 1 이 아니고 0 입니다.) 부터
2 번째 전까지 잘라내라 }

라는 뜻입니다.

0을 생략할 수 있어서 이렇게 써도 작동합니다.

```
[32]: 문자열[:2]
```

```
[32]: '안녕'
```

결론

사용하는 형식이 약간 다를 뿐이지, 엑셀의 함수와 파이썬 언어는 본질적으로 같은 일을 하기 위해 필요한 재료의 개수도 같고, 그 입력 형식도 많이 다르지 않습니다. 익숙하냐 그렇지 않냐 의 차이일 뿐이라고 생각합니다.

물론 그럼 익숙한 엑셀 쓰지 왜 파이썬을 쓰나요? 라는 질문이 나올 텐데, 파이썬으로 엑셀 보다 훨씬 더 다양하고 유연하게 상황을 해결할 수 있다고 일단은 대답할 수 있겠네요.

어쨌든 이 글은, 프로그래밍이 너무 어렵게 보여서 망설이던 사람들을 파이썬으로 유도하는 성격의 글이라고 봐 주시면 감사하겠습니다. 2023 심화세미나의 목표 중 하나로, 팀원들에게 파이썬 진입장벽을 낮추는 이벤트 강연이 있었는데, 이렇게 문서를 작성해서 공유하는 것으로 계획을 바꾸었습니다.

익숙한 엑셀 함수를 동일하게 파이썬에서도 작성해보면서 좀 친근하게 느껴보자는 게 취지이고요.

하반기에는 파이썬 튜토리얼도 작성해서, 관심 생긴 분들이 쉽게 초반 러닝 커브를 극복하는데 조금이라도 도움될 수 있도록 할 예정입니다.

※추가 설명.

Q>

Python으로 뭘 하려면,
이렇게 데이터를 Typing으로 입력해야
하나?

A>

당연히 그럴 리 없고, 엑셀 파일을 읽어들
여서, 자동으로 스프레드 데이터를 리스트
의 형식으로 가져오게 됩니다.

설명의 군더더기를 없애기 위해,
데이터를 타이핑으로 직접 선언했습니다.