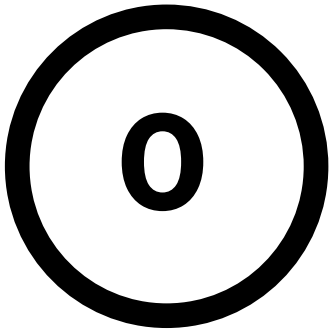
파이썬을 활용한
데이터 분석 과정

파이썬 기초 핵심 문법부터 데이터분석 & 머신러닝 기반 분석 방법론까지

수업을 시작하기 전에

모든 자료는 구글 드라이브에 공유됩니다.구글 계정이 있으셔야 합니다.

01 데이터분석을 위한 파이썬 프로그래밍



데이터분석 어떻게 시작할까?

01

데이터분석을 하려면 공부해야하는 것들

❖ 데이터분석 관련 필요 역량

프로그래밍	데이터분석이론	수학	도메인 지식
파이썬	전처리 방법론	선형대수학	금융?의료?법률? ...
C	피처 엔지니어링	미적분	
JAVA	머신러닝 분석 이론	확률	데이터 이해력
Julia	통계기반 분석 이론	통계	분석 설계 능력
:	:	:	결과 분석력
SQL			:
Linux			
Docker			
VM			
:			

01

데이터분석을 하려면 공부해야하는 것들

❖ 데이터분석 관련 필요 역량

프로그래밍
C
JAVA
Julia
:
SQL
Linux
Docker
VM
:

데이터분석이론
피처 엔지니어링
머신러닝 분석 이론
:

수학
미적분
확률
:

도메인 지식
데이터 이해력
분석 설계 능력
결과 분석력
:

갖춰야할 역량이 많다..

컴공/수학/통계 혹은 유사전공 중 어느쪽도 아닌 완전 비전공자라면 포기하고 싶을 정도..?

그럼 이제, 아래 선택지 중 하나를 선택하자.

1. **포기하고 다른 길을 모색하자.**(슬라이드를 닫으세요.)
2. **쉬운 것부터 일단 시작해보자.**(다음 슬라이드로 이동)

01

데이터분석을 하려면 공부해야하는 것들

❖ 데이터분석 관련 필요 역량

프로그래밍	데이터분석이론	수학	도메인 지식
파이썬	전처리 방법론	선형대수학	금융?의료?법률? ...
C	피처 엔지니어링	미적분	
JAVA	머신러닝 분석 이론	확률	데이터 이해력
Julia	통계기반 분석 이론	통계	분석 설계 능력
:	:	:	결과 분석력
SQL			:
Linux			
Docker			
VM			
:			

가장 쉬운거.. 그나마 쉬운거를 찾아보자.. 쉬운거.. 쉬운거..
잘 모르겠다는 분을 위해 추천해드리자면..

01

데이터분석을 하려면 공부해야하는 것들

❖ 데이터분석 관련 필요 역량

프로그래밍	데이터분석이론	수학	도메인 지식
파이썬	전처리 방법론	선형대수학	금융?의료?법률? ...
C	피처 엔지니어링	미적분	
JAVA	머신러닝 분석 이론	확률	데이터 이해력
Julia	통계기반 분석 이론	통계	분석 설계 능력
:	:	:	결과 분석력
SQL			:
Linux			
Docker			
VM			
:			

상.대.적.으.로 가장 쉽게 눈에 보이는 결과물을 만들어 낼 수 있다.
결과물을 만들어낼 수 있다보니, 조.금 재밌다.

01

그나마 할만한 파이썬

프로필

이름(한글) : 파이썬　(영어) Python

출생연도 : 1991년　　가족관계 : 귀도 반 로섬(아버지)

비전 : 개발자들 칼퇴시키기/ 명절에 가족들과 시간보내게 하기

강점 : 1. 오픈소스가 방대하다. / 2. 코드가 읽고 쓰기 쉽다.

01

그나마 할만한 파이썬

프로필

이름(한글) : 파이썬　(영어) Python

출생연도 : 1991년　　가족관계 : 귀도 반 로섬(아버지)

비전 : 개발자들 칼퇴시키기/ 명절에 가족들과 시간보내게 하기

강점 : 1. 오픈소스가 방대하다. / 2. 코드가 읽고 쓰기 쉽다.

01

그나마 할만한 파이썬

❖ 오픈소스! 이곳은 보기보다 아름다운 세상

내가 한달간 밤을 새가면서 열심히 만든 작품이 있는데, 그 작품을 필요한 익명의 다수에게 '**무상으로**' 제공할 수 있을까?

오픈소스는 누구나 열람하고 사용할 수 있도록 오픈된 코드라고 보면 된다.

오픈소스 가운데, 파이썬으로 짜여진 코드가 매.우. 상당히 많은 것으로 알려져 있다.

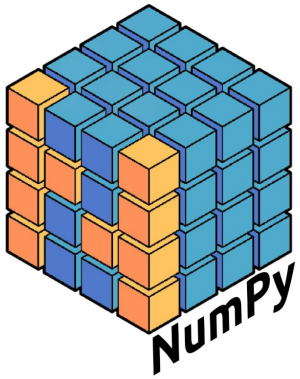
머신러닝 관련 파이썬 오픈소스도 참 많다. 오픈소스를 활용해, 손쉽게 머신러닝 분석에 입문이 가능하다.

Thanks to Open-Source

02

앞으로 배울 데이터분석 관련 오픈소스

Beautiful Soup



matplotlib

Seaborn

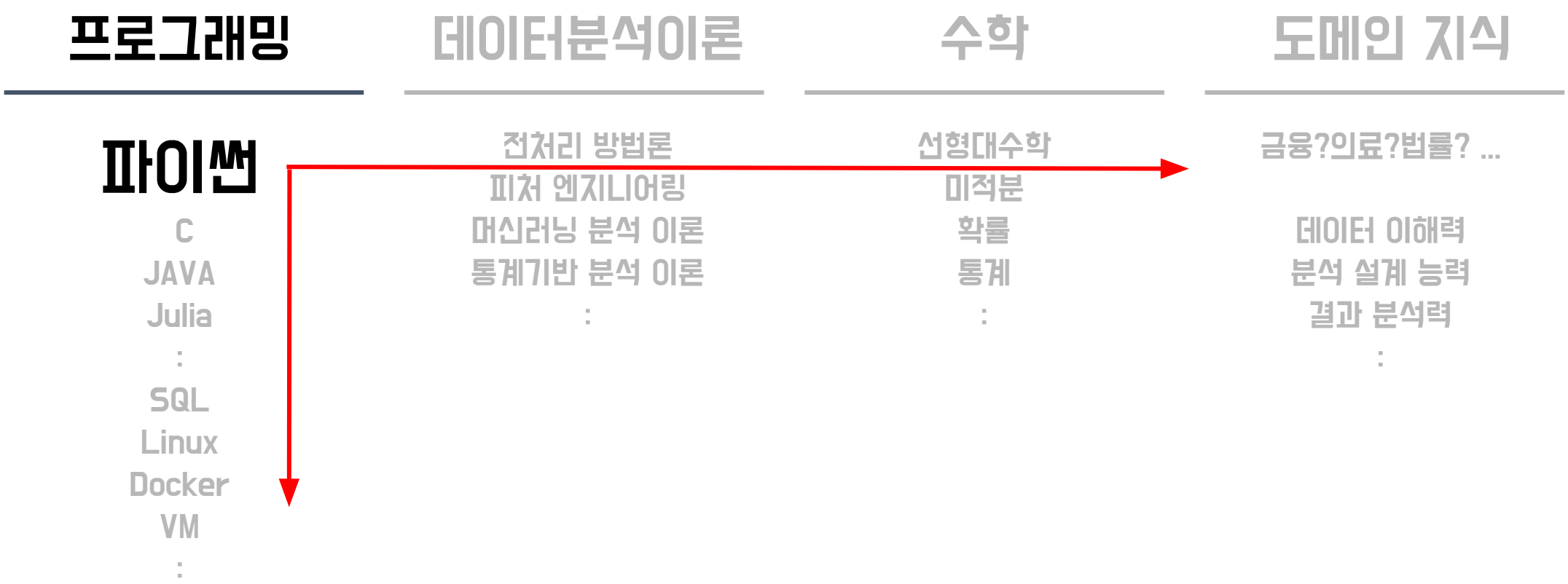
Pandas

TensorFlow

02

데이터분석 로드맵

❖ 데이터분석 추천 공부 순서



03

실습 환경은 Colab

Colab 은 구글에서 제공해주는 서비스

구글의 서버를 지원

- 컴퓨터 사양이 일반 개인 PC보다 좋다.
- 파이썬과 웬만한 패키지가 모두 설치되어있다.(설치필요X)

04

Colab Notebook 사용법 익히기

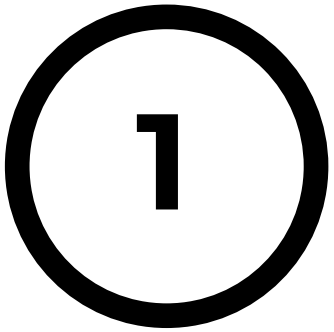
Colab Notebook은 자체 단축키가 있다.

단축키를 미리 익혀두면 사용하는데에 매우 편하다.

핵심적인 개념은 노트북은 Cell로 이루어져 있다는 것!

- Cell의 종류 : Code Cell 과 Markdown Cell이 있다.
- Mode : 커맨드 모드와 에디트 모드가 있다.

각각에 맞는 단축키를 숙지해두자.



연산(Operator)

생각1. 10을 3으로 나눈 몫과 나머지를 구하고 싶다면?

생각2. 100,000,000과 같은 큰 수를 간단하게 표현하고 싶다면?

생각3. 5 > 3의 결과는 뭘까?

생각4. '같다'와 '같지않다'는 어떻게 표현할까?

변수(Variable)

생각1.

23412 * 2, 23412 * 3, 23412 * 4 를 계산해보자.

이번엔, 34422 * 2, 34422 * 3, 34422 * 4 를 계산해보자.

이번엔, 6575 *2, 6575 *3, 6575 *4 를 계산해보자.

이번엔, ...

(그만..) 이라는 생각이 들기 마련이다.

데이터를 재사용할 수 있으면 얼마나 좋을까?

생각2.

변수는 여러 타입이 있다. 정수, 실수, 문자열, Boolean ...

타입이 궁금하다면?

생각3.

문자열은 ' ' 혹은 " "로 문자열을 감싸줘야 한다. 둘의 차이는 없다.

이건 어떤 장점이 있을까?

친구를 만나느라 "샤샤샤" 를 출력해보자.

생각4.

"올라프"는 '정말' 귀여워 를 출력하고 싶다면?

생각5.

님은 갔습니다.

아아. 사랑하는 나의 님은 갔습니다.

를 출력하고 싶다면?

생각6.

var = 4 의 타입을 문자열로 바꾸고 싶다면?

① str(var)

② var = str(var)

생각7.

다음의 결과는 어떻게 될지 생각해보자.

bool(123) →

bool(-123) →

bool(0) →

bool('엘사') →

bool(' ') →

bool('0') →

bool('') →

자료구조(Data Structure)

리스트

생각1.

toy1 = 우디

toy2 = 버즈

toy3 = 포키

toy4 = 보

...

하나의 변수 안에 여러개의 데이터를 넣을 수 없을까?

toy = [우디, 버즈, 포키, 보, ...]

리스트 - 인덱싱/슬라이싱

생각2.

자료구조 첫번째. 리스트

toy = [우디. 버즈. 포키. 보. ...]

버즈만 지칭하고 싶다면?

생각3.

우디부터 포키까지 지칭하고 싶다면?

리스트 - 데이터 추가

생각4.

toy = [우디. 버즈. 포키. 보]
toy에 새로운 장난감 제시를 추가 하고 싶다면?

생각5.
슬링키독을 포키 앞에 추가(삽입)하고 싶다면?

생각6.
포테이토헤드와 햄을 한번에 추가하고 싶다면?

리스트 - 데이터 삭제

생각7.

toy =

['우디', '버즈', '보', '슬링키독', '포키', '제시', '포테이토 헤드', '햄', ['포테이토 헤드', '햄']]

toy에서 잘못된 형식으로 추가된 ['포테이토 헤드', '햄']을 제거하고 싶다면?

생각8.

데이터를 지칭하지 않고, 그냥 마지막 데이터를 뽑아내고 싶다면?

생각9.

인덱스를 지칭해서 해당 인덱스 자리의 데이터를 뽑아낼 수도 있을까?

리스트 - 데이터/변수 삭제

생각10.

toy = [우디, 버즈, 슬링키독, 포키]

toy 안에 있는 모든 데이터를 한번에 삭제하고 싶다면?

생각11.

toy 변수까지 삭제하고 싶다면?

리스트 - 정렬 / 각종 정보 조회

생각12.
리스트 안에 데이터를 일정 기준으로 정렬을 하고 싶다면?

생각13. 리스트 안에 데이터가 몇 개 들어있을까?

생각14. 아이언맨은 몇번째 자리에 있을까?

생각15. 아이언맨은 몇개있을까?

생각16. 앤트맨은 HERO 변수안에 있을까?

튜플

생각1.
리스트 안의 데이터를 더이상 변경시키고 싶지 않다면?

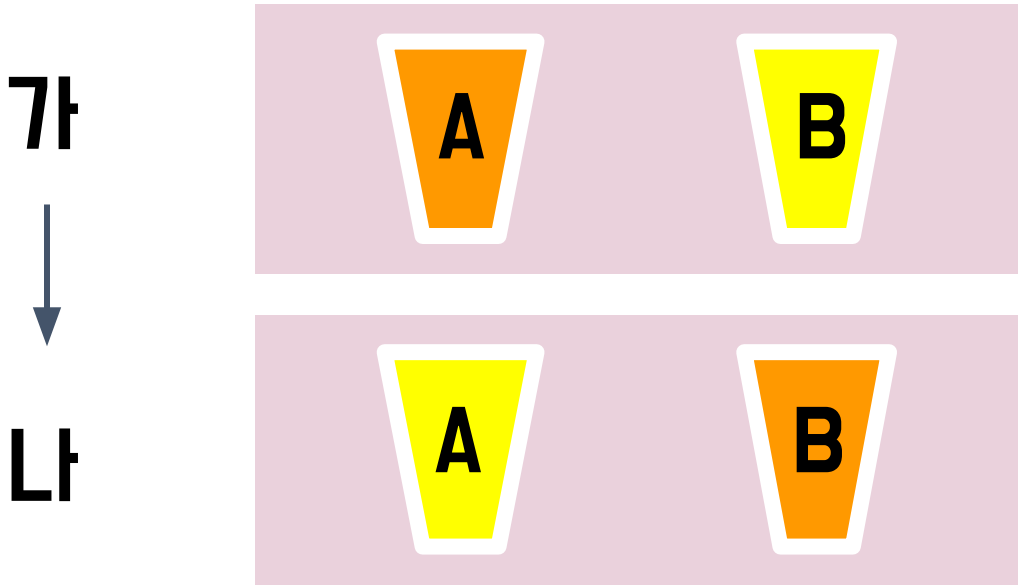
생각2.
리스트에서 사용하던 추가/삭제 기능을 사용해보면?

생각3.
그래도 튜플에 데이터를 추가하고 싶다면?

생각4.
리스트에서 사용하던 기능들 중 똑같이 사용할 수 있는 것들이 있을까?

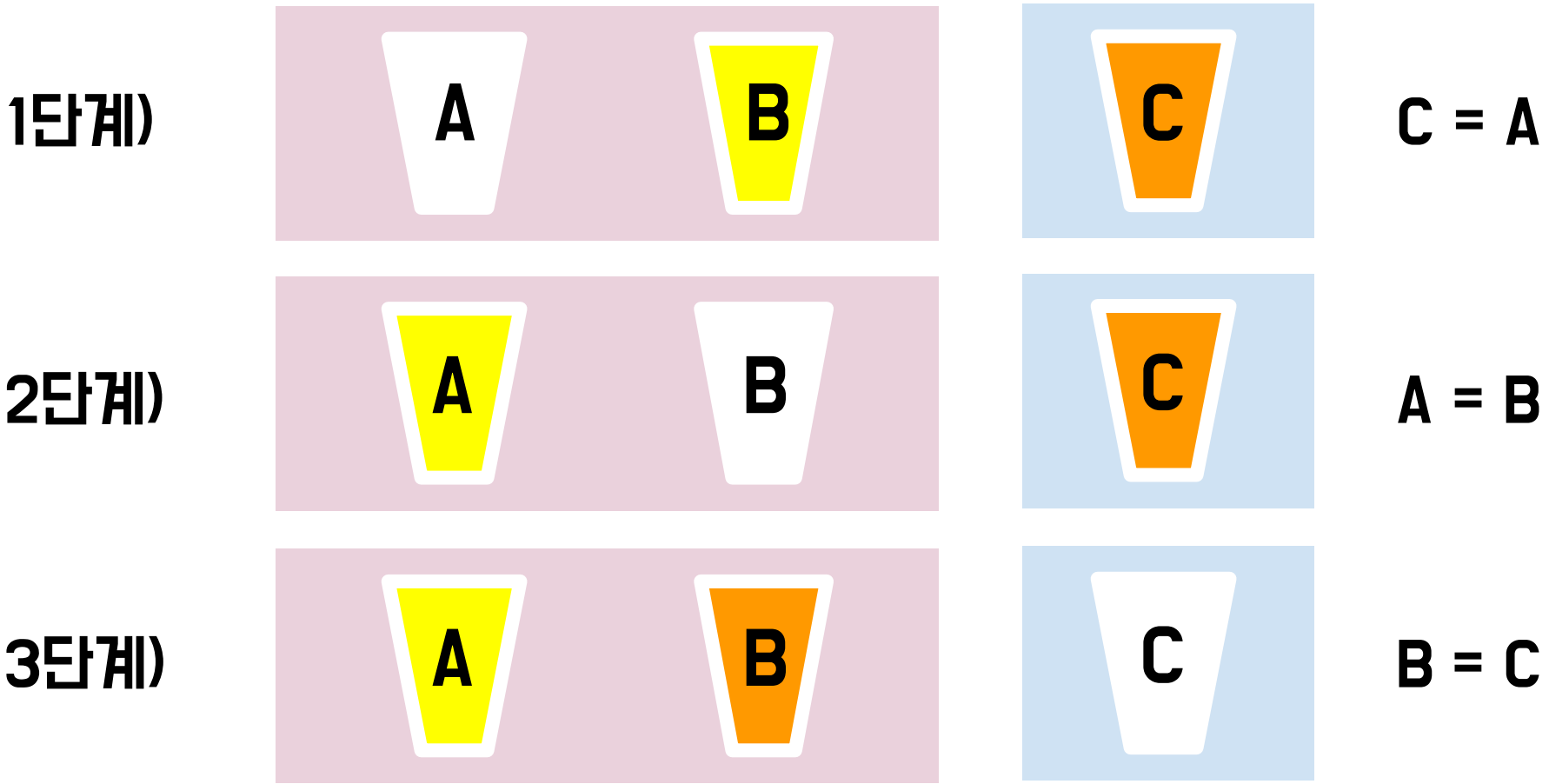
튜플

생각5.
아래와 같이 두 컵에 두 종류의 음료가 있다. 양쪽의 음료를 아래그림 같이 옮겨 담고 싶다면?



튜플

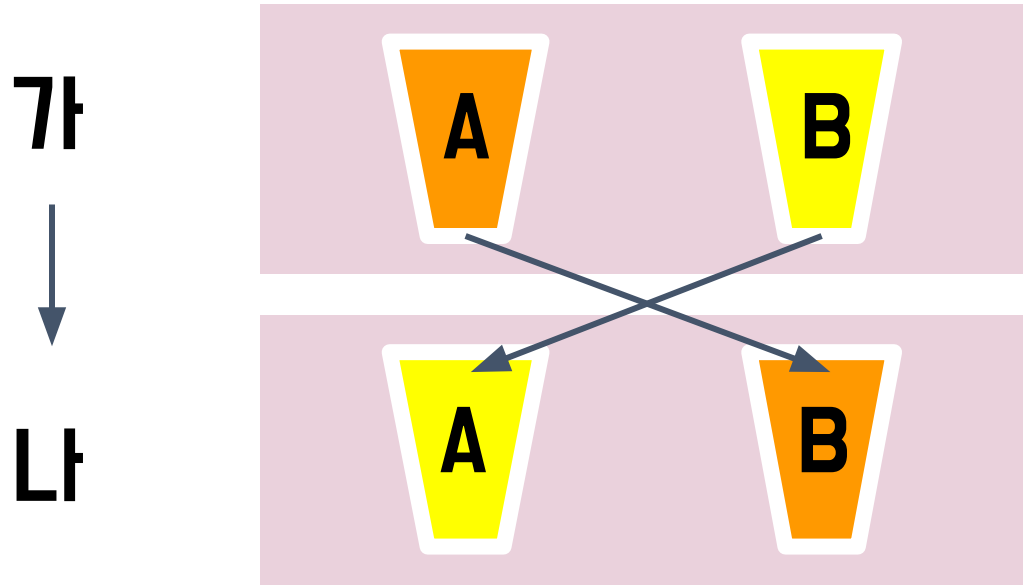
1단계) 빈컵(C) ← A ⇒ 2단계) A ← B ⇒ 3단계) B ← C



튜플

생각5.
하지만!! 파이썬은 그렇게 할 필요가 없다! 이것이 해체할당기능의 매력!

가



나

A, B = B, A

집합(셋)

생각1.
리스트 안의 중복된 데이터를 제거하고 싶다면?

생각2.
집합에서는 어떤 연산 기능들이 있을까?

딕셔너리

생각1.
이름 : 알디노. 직업 : [개발자. 강사] 이런 식으로 key값과 value값을 함께 담고 싶다면?

생각2.
딕셔너리에서 알디노 값을 조회하고 싶다면?

생각3.
전공분야 : 전자전기공학 을 추가하고 싶다면?

딕셔너리

생각4.
전공분야 의 값을 머신러닝으로 바꾸고 싶다면?

생각5.
딕셔너리에서 데이터를 삭제하려면?

생각6.
딕셔너리에 특정 데이터가 있는지 궁금하다면?
per_info에 '이름'이라는 값이 있을까?

조건문(If, else, elif)

조건문

생각1.
특정 조건을 만족할 경우에만 수행되어야 하는 경우에는 어떻게 해야할까?

예를 들어.
영화를 예매하는 상황을 가정해보자.
보고싶은 영화를 클릭했다. 잔여석이 있다면, 티켓 안내문구가 나와야 하지만 잔여석이 없다면 티켓 안내문구가 나와서는 안된다.
이런 프로그램을 작성할 때는 어떻게 해야할까?

조건문

생각2.
특정 조건을 만족하지 못할 경우에만 수행되어야 하는 경우에는?

이전과 같은 가정에서.
보고싶은 영화를 클릭했다. 잔여석이 있다면, 티켓 안내문구가 나오고.
잔여석이 없다면 잔여석이 없다는 문구가 나와야 한다.
이런 프로그램을 작성할 때는 어떻게 해야할까?

조건문

생각3.
특정 조건을 만족하지 못하지만, 다른 조건은 만족하는 경우에는?

이전과 같은 가정에서,
보고 싶은 영화가 2개가 있다. 먼저 보고 싶은 영화를 먼저 잔여석이 있는지 확인하고, 잔여석이 없다면 다음 영화의 잔여석을 확인해주도록 만들고 싶다면?

반복문(for, while)

반복문

생각1.
동일한 작업을 여러번 반복하고 싶은 코드를 작성해야 한다면?

생각2.
for i in range(5): → 이 문장의 의미는 뭘까?

반복문

생각3.
i 가 1부터 2씩 증가해서 9가 될때까지 반복하는 반복문을 만들고 싶다면?

생각4.
아래와 같이 출력하는 코드를 반복문을 이용해서 작성해보자.
결과 :

```
    *
   * *
  * * *
 * * * *
* * * * *
```

반복문

생각5.
for i in 리스트:
의 단점은 반복문이 몇 번째 loop인지 직관적으로 알기 어렵다는 것이다.
이 단점을 극복하려면 어떤 정보를 조회하면 될까?

생각6.
여러개의 리스트를 동시에 반복문으로 다루고 싶다면?

반복문

생각7.
무한반복을 돌게 할 수는 없을까?

생각8.
그럼, 특정 조건이 충족될 때까지만 반복하게 할 수도 있을까?

반복문

생각9.
반복문 안에서 또 다른 반복을 시행해야 하는 경우는 없을까?

반복문

조건문과 반복문에서의 핵심은 **"규칙의 단순화"**라고 할 수 있다.

풀어 설명하자면,

1. 규칙을 발견하고,
2. 이 규칙을 단순하게 표현하는 방법을 구상해서,
3. 코드로 작성한다.

그렇다. 이 부분은 암기로 해결할 수 없는 **'논리적 사고력'**의 영역이다.
이는 끊임없는 훈련이 필요하다. 머리로만 생각하지말고,
반드시 **쓰면서 생각을 정리하는 연습**을 하자.

화면 입출력(Screen I/O)

input, format

화면 입출력

생각1.
변수에 넣을 데이터를 실시간으로 직접 입력받고 싶다면?

생각2.
숫자형 데이터를 입력받고 싶다면?

화면 입출력

생각3.

출력할 때, 짜임새있게 형식을 지정해서 출력하고 싶다면?

화면 입출력

생각4.
아래 (A)와 (B)의 차이는 뭘까? (B)와 같이 출력하려면?

```
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
21 22 23 24 25 26 27 28 29 30
31 32 33 34 35 36 37 38 39 40
41 42 43 44 45 46 47 48 49 50
51 52 53 54 55 56 57 58 59 60
61 62 63 64 65 66 67 68 69 70
71 72 73 74 75 76 77 78 79 80
81 82 83 84 85 86 87 88 89 90
91 92 93 94 95 96 97 98 99 100
```

(A)

```
  1   2   3   4   5   6   7   8   9  10
 11  12  13  14  15  16  17  18  19  20
 21  22  23  24  25  26  27  28  29  30
 31  32  33  34  35  36  37  38  39  40
 41  42  43  44  45  46  47  48  49  50
 51  52  53  54  55  56  57  58  59  60
 61  62  63  64  65  66  67  68  69  70
 71  72  73  74  75  76  77  78  79  80
 81  82  83  84  85  86  87  88  89  90
 91  92  93  94  95  96  97  98  99 100
```

(B)

화면 입출력

생각5.
아래 (C) 처럼 출력하고 싶으면? (D) 처럼은?

```
001 002 003 004 005 006 007 008 009 010
011 012 013 014 015 016 017 018 019 020
021 022 023 024 025 026 027 028 029 030
031 032 033 034 035 036 037 038 039 040
041 042 043 044 045 046 047 048 049 050
051 052 053 054 055 056 057 058 059 060
061 062 063 064 065 066 067 068 069 070
071 072 073 074 075 076 077 078 079 080
081 082 083 084 085 086 087 088 089 090
091 092 093 094 095 096 097 098 099 100
```

(C)

```
1   2   3   4   5   6   7   8   9   10
11  12  13  14  15  16  17  18  19  20
21  22  23  24  25  26  27  28  29  30
31  32  33  34  35  36  37  38  39  40
41  42  43  44  45  46  47  48  49  50
51  52  53  54  55  56  57  58  59  60
61  62  63  64  65  66  67  68  69  70
71  72  73  74  75  76  77  78  79  80
81  82  83  84  85  86  87  88  89  90
91  92  93  94  95  96  97  98  99  100
```

(D)

화면 입출력

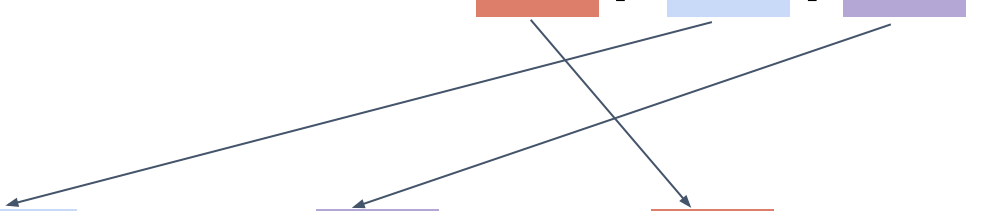
생각6.
출력할 대상이 실수(float)일때는. 소숫점 자리가 문제다. 이 또한 형식을 지정해주려면?

쉽게 말해.
a = 1.33333 인 경우에. 소숫점 2째자리까지만 출력하고 싶다면?

화면 입출력

생각7.
형식지정출력을 사용할 때, 자리를 지정해주려면?

예를 들어,
'1st : {} , 2nd : {}, 3rd : {}'.format('우디', '버즈', 포키')



결과 → 1st : 버즈 , 2nd : 포키, 3rd : 우디

07 함수(Function)

함수

코드는 단순화시키자면, 변수와 함수로 이루어져 있다고 볼 수 있다.

변수는 이미 배웠다. 변수를 사용하는 이유를 다시 상기해보자.

데이터의 재사용과 효율적인 관리를 위해서였다.

함수도 마찬가지다. 함수를 사용하는 이유는
코드의 재사용과 효율적인 관리를 위해서라고 정리할 수 있다.

함수

변수에 데이터를 넣어두면, 변수명을 호출함으로써 데이터를 불러올 수 있다.

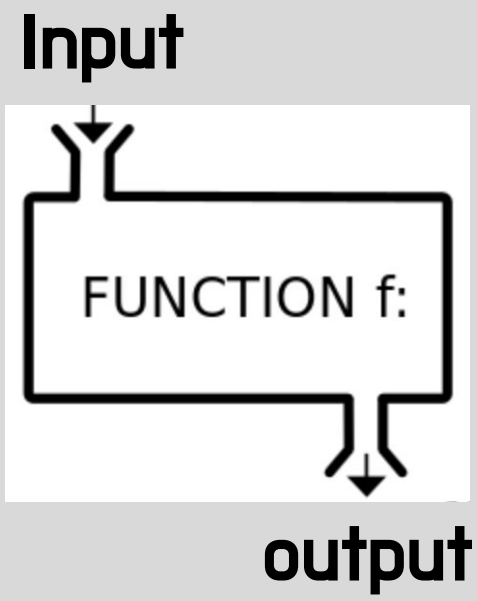
마찬가지로 다음과 같이 함수에 코드를 넣어두고, 함수명을 호출함으로써 코드를 불러올 수 있다.

```
def 함수명():     #함수 정의
    ------
     코 드
    ------
```

```
함수명()          #함수 호출
```

함수

우리는 함수라는 개념을 수학시간에 배운 적이 있다. 아래 그림을 보자.



코드를 담고 있는 함수에도 Input과 Ouput이 각각 필요한 경우가 있고, 필요가 없는 경우가 있다. 이에 따라 4가지 종류로 구분된다.

함수

4가지 종류는 다음과 같이 형태와 목적이 다르다.

입력x, 출력x

```
def 함수명():
    ------
     A코드
    ------
```

```
함수명()
```

단순 A코드 재사용

입력o, 출력x

```
def 함수명(Input):
    ------
     A코드
    ------
```

```
함수명(Input)
```

A코드 변형해서 재사용

입력x, 출력o

```
def 함수명():
    ------
     A코드
    ------
    return output
```

```
함수명()
```

함수명() ← output
output 재사용

입력o, 출력o

```
def 함수명(Input):
    ------
     A코드
    ------
    return output
```

```
함수명(Input)
```

A코드 변형해서 재사용
+
함수명() ← output
output 재사용

함수

생각1.
함수를 호출할 때, 함수의 입력 인자 개수를 맞춰주지 않으면 어떻게 될까?

예를 들어,
def create_info_dict(name, phone, email) ← 입력인자 3개

create_info_dict('이원재') → 1개

함수

생각2.
에러가 난다. 그럼 함수를 사용하려면, 넣어줘야 할 입력인자를 모두 알고 있어야할까?

예시)
def create_info_dict(name, phone, email) ← 입력인자 3개

create_info_dict('이원재') → 1개

함수

생각3.

입력인자 개수를 상황에 따라 가변적으로 조절해도 되는 함수를 만들려면 어떻게 해야할까?

생각4.

여러개의 데이터를 리스트로 묶어서 넣어주면 어떻게 될까?

함수

생각5.

가변인자를 key값을 가진 딕셔너리 형태의 변수로 선언하고 싶다면?

생각6.

넣어줄 때도 딕셔너리로 묶어서 넣어주면 어떻게 될까?

08 내포(Comprehension)

내포

생각1.

[0, 1, 2, . . . , 97, 98, 99]

이런 리스트를 만들어보자. 반복문을 써야 한다. 그런데.. 고작 리스트를 담는 변수를 하나 선언하는데에 몇 줄에 걸쳐 코드를 작성해야한다니.. 훨씬 수월한 방법은 없을까?

내포

생각2.
조건문 if else 도 내포가 가능할까?

생각3.
elif는 어떨까?

내포

생각4.
딕셔너리도 내포 용법으로 선언할 수 있을까? 어떻게 하면 될까?

09 클래스(Class)

클래스

변수&함수 Review

"코드는 단순화시키자면, 변수와 함수로 이루어져 있다고 볼 수 있다."
"변수 → 데이터 재사용&관리 / 함수 → 코드 재사용&관리"

객체(Object)의 개념

"파이썬은 객체들의 세상이다"
객체 ← 데이터, 변수, 함수

클래스(Class)의 개념

"클래스는 객체를 만드는 설계도(틀) or 공장이라고 할 수 있다"
→ 객체를 재사용&관리 하기 위해서 사용한다고 볼 수 있다.

클래스

객체 개념 한단계 UP

"파이썬은 객체들의 세상이다"
객체 ← 데이터, 변수, 함수

그럼, 게임을 만든다고 해보자.
여러 유닛을 만들고 싶은데, 유닛들마다 공통적으로 갖고 있는 속성과 행위 요소가 있을 수 있다. (예를 들어, 속성 = 공격력, 방어력/ 행위 = 공격하기, 방어하기 등)

→ 속성(변수) 와 행위(함수)를 갖고 있는 유닛(새로운 변수/객체)를 만들고 싶다.

이럴 때 클래스(설계도, 공장)을 만들어놓고, 클래스로 여러 객체를 만들어내면 된다.

→ 여기서 클래스로 만들어낸 객체를 인스턴스(instance)라고도 부른다.

클래스

"객체(인스턴스) ← 클래스" 라는 개념

1) 자전거A, 자전거B, 자전거C 를 만들고 싶다. → 자전거 A, B, C = 객체

2) 자전거를 만들 설계도 or 공장이 필요하다. → 자전거 설계도 = 클래스

3) 설계도 or 공장을 먼저 만들어야 한다.

4) 자전거를 만들 설계도/공장에서 설계해줘야 하는 요소는 다음 2가지다.
   자전거의 "속성"과 "행위"

5) 설계도 or 공장을 다 만들었으면, 자전거를 찍어내면 된다.

클래스

"객체(인스턴스) ← 클래스" 라는 개념

클래스 선언

```
class 자전거():
    자전거 "속성" 결정

    자전거 "행위" 결정
```

객체 선언 ← 클래스 호출

```
자전거A = 자전거()
```

클래스

동의어 정리만 잘해도 클래스는 쉽다

객체 = 인스턴스 = self

인스턴스의 속성 = 변수 = 초기화 함수(__init__())

인스턴스의 행위 = 함수

```
class Unit():
    def __init__():
        self.공격력 =

    def 공격하기():
```

클래스

실습1.

자전거 클래스를 만들어보자.

자전거의 속성 : 바퀴 크기와 바디 색 / 자전거의 행위 : move() turn() stop()

실습2.

자전거 클래스를 가지고, 삼천리 인스턴스를 만들어보자.

삼천리의 속성 : 바퀴 크기 = 200, 바디 색 = 노랑

클래스

실습3.
클래스 안에서 instance_count라는 변수를 만들고,
클래스로 인스턴스를 만들 때마다 instance_count의 값을 증가시켜서 클래스로 만든 인스턴스의 개수를 세보자.

실습4.
instance_count 를 출력해주는 count_instance()라는 함수도 만들어보자.

클래스

생각1.
자전거라는 클래스를 이미 만들어놨다.
이번엔 접이식 자전거 클래스를 만들고 싶다.

접이식 자전거의 속성과 행위 중 기본적인 것은
이미 만들어놓은 자전거 클래스와 겹친다.

이런 경우에 겹치는 것을 그대로 가져다 쓸 수 있으면 편하지 않을까?

10 모듈(Module)과 패키지(Package)

모듈과 패키지

생각1.
파일을 하나 생성해서 열심히 코드를 작성했다.
이 코드를 다른 파일에서도 그대로 가져다 쓸 수 있으면 얼마나 편할까?

모듈과 패키지

생각2.
모듈을 재사용하려면, 모듈이 잘 돌아가는지 검증해주는 코드도 있어야 한다. 모듈을 검증할때는 검증코드가 활성화되고, 다른 파일에서 해당 모듈을 재사용할 때는 검증코드가 활성화되지 않도록 코드를 작성하고 싶다면?

모듈과 패키지

생각3.
같은 카테고리로 묶을만한 모듈이 여럿 존재한다면?
같은 카테고리의 파일을 한 폴더로 관리하듯이 모듈도 이와같이 폴더로 관리하고 싶다면?

11 크롤링 기초

크롤링 기초

생각1.
크롤링을 알려면, 먼저 HTML을 알아야 한다.
HTML이 뭔데?

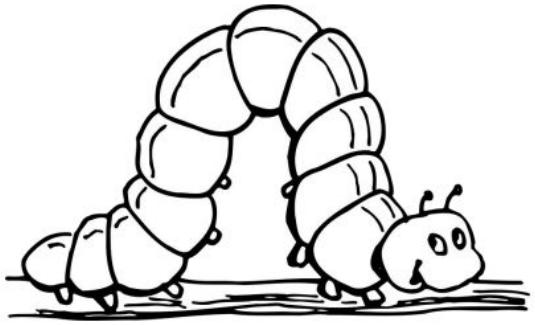
크롤링 기초

HTML의 정의
"HTML == 웹 상의 데이터"

HTML의 3가지 진실
1. HTML의 기본 단위는 태그(tag)다.
2. 태그 안에 웹 상의 데이터가 들어가 있다.
3. HTML은 태그의 계층적인 구조로 이루어진 "태그들의 계층 사회"다

크롤링 기초

생각2.
크롤링이 뭘까?



크롤링 기초

크롤링의 정의

크롤링은 웹 상에서 원하는 데이터를 가져오는 행위를 말한다.

〈정리〉

크롤링이란

웹 상에서

→ 원하는 HTML 안에서

→ 원하는 태그를 찾는 것이다.

: 크롤링은 (태.그.찾.기)다.

크롤링 기초

생각3.
원하는 데이터를 담고있는 태그를 어떻게 찾지?

크롤링 기초

생각3 (정리 버전)

원하는 데이터를 담고있는 태그를 어떻게 찾지?

→ 인터넷으로 연결되어 있는 전세계의 웹 상의 무수히 많은 HTML 중에서

원하는 HTML은 어떻게 찾으며. (→ 1단계)

원하는 HTML을 찾는다해도. 태그의 계층 사회인 HTML 속에서

원하는 태그를 어떻게 찾지? (→ 2단계)

크롤링 기초

크롤링하는 법

1단계. 원하는 HTML 찾기 → URL 로 찾는다. → requests → requests.get(url)

2단계. 원하는 태그 찾기 → 수식어로 찾는다. → beautifulsoup → find / findall / select

수식어 1. 태그의 계층 구조 (예. 서연이 아들 → 민우의 아들 → 민준이 찾아줘)

수식어 2. 태그의 속성 (예. 전라도 고창에 사는 민준이 찾아줘)

→ 수식어 1 + 수식어 2 함께 이용해 찾는다.

→ 유니크한 속성을 가진 조상님 태그를 이용하는 것이 핵심이다.

크롤링 기초

생각4.
URL이 너무 긴경우가 있던데?
자세히 보면, 쿼리스트링 부분이 너무 길다. 이럴 땐 어떻게 할까?

크롤링 기초

생각5.
이미지를 크롤링하고 싶다면?
지금까지 원하는 데이터는 HTML 에 모두 보여져 있었기에, beautifulsoup으로 파싱이 가능했다. 하지만, 이미지는 HTML에 보이지 않는다. 어떻게 해야할까?

파일 입출력

파일 읽고 쓰기

```
with open('파일경로/파일명.확장자명', '파일입출력형식') as f:
    f.파일입출력함수()
```

예) data폴더에 있는 img0.jpg파일을 읽어들이고 싶다!

```
with open('data/img0.jpg', 'rb') as f:
    f.read()
```

data폴더에 있는 img1.jpg파일을 쓰고 싶다!

```
with open('data/img1.jpg', 'wb') as f:
    f.write()
```

파일 입출력

파일입출력형식 정리

	텍스트 파일	바이너리 파일
읽기	'rt' ('r'만 쓰거나 안써도 됨)	'rb'
쓰기	'wt' ('w'만 써도 됨)	'wb'

파일 입출력

파일입출력형식 정리 (기본)

	텍스트 파일	바이너리 파일
읽기	1) read() : 파일내용 하나의 문자열로 한번에 읽어들이기 2) readline() : 첫 한 줄만 읽어들이기 3) readlines() : 한줄씩 파일내용 전부 읽어들이기(한줄씩, 리스트로 저장)	read() : 파일내용 읽어들이기
쓰기	1) write() : 하나의 문자열로 파일에 쓰기 2) writelines() : 문자열리스트로 파일에 쓰기	write() : 파일에 쓰기

크롤링 기초

생각6.
iframe 의 경우, 어떻게 크롤링할 수 있을까?

생각7.
JSON은 HTML과 어떻게 다를까? 크롤링하는 방법도 다를까?

12 정규표현식

정규표현식

정규표현식이 뭡가요?

영어로 Regular Expression. 특정 규칙을 가진 문자열을 표현하는 방식이다.

정규표현식의 용도

문자열 가운데서 특정 문자열만 찾아내고 싶을 때. 찾고싶은 대상을 정규표현식으로 찾아낸다.

정규표현식의 사용법

1) import re
2) re.findall('찾고싶은 대상→ 정규표현식으로 표현' , 찾을 위치)

정규표현식

대표 문법 10가지

찾고싶은 대상을 정규표현식으로 표현하면 되는데. 알아야할 정규표현식은 아래 10개면 충분하다.

1	.	모든 문자 1개	6	[a-z]	a부터 z까지 중 하나
2	a?	a가 0회 또는 1회	7	[^가-힣]	가부터 힣까지 제외한 것들 중 하나
3	a*	a가 0회 이상	8	\+	회피 용법. 특수기호로써의 +말고 진짜 +
4	a+	a가 1회 이상	9	.+	문자 여러개 - Greedy Quantifier
5	a(b)c	a 뒤. c 앞에 있는 b / abc에서 b만	10	.+?	문자 여러개 - Reluctant Quantifier

정규표현식

생각1.

"서울대학교(영어: Seoul National University)[4]는 대한민국의 국립대학이다."
에서 'Seoul National University' 만 찾고 싶다면?

생각2.

"서울대학교(영어: Seoul National University)[4]는 대한민국의 국립대학이다."
에서 '서울대학교' 만 찾고 싶다면?

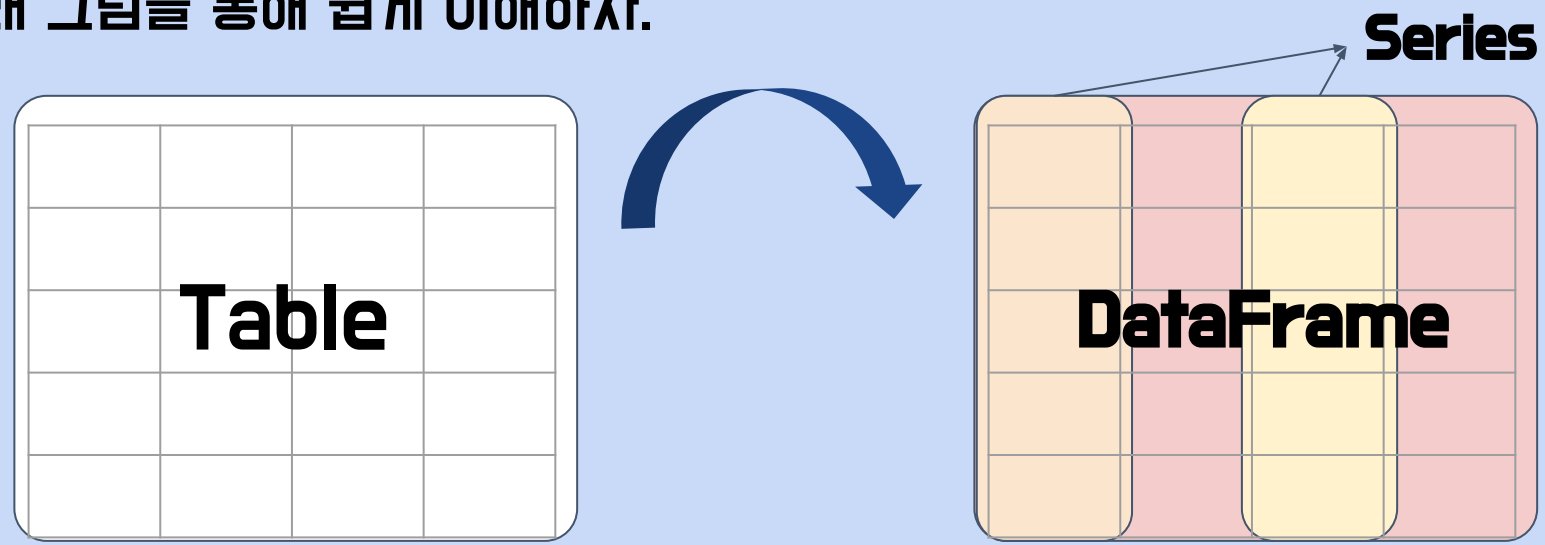
Pandas

Pandas

Pandas란?

Pandas 패키지는 파이썬으로 데이터 분석을 할 때, 가장 많이 쓰이는 패키지다.
Pandas는 정형데이터 즉, Table 데이터(더 쉽게 말해, 엑셀로 정리되는 데이터)를
Series와 DataFrame이라는 Type으로 다루는 패키지다.

아래 그림을 통해 쉽게 이해하자.



Pandas

1) dataset = pd.read_csv(file_path) → file 불러오기

2) dataset.head() / dataset.tail() → data 상단/하단 조회

3) dataset.loc[] → data 일부 행 조회

4) dataset[열이름_리스트] → data 일부 열 조회

* dataset 열 순서 바꾸고 싶다면? 열 일부 삭제하고 싶다면?

→ 모두 열이름_리스트를 다뤄 바꾼다.

Pandas

5) dataset.rename(columns={기존이름:바꿀이름}) → 열이름 수정

6) dataset.drop([행이름1, ...]) → 행 삭제(1개 또는 여러개)

7) pd.concat([데이터프레임1, 데이터프레임2]) → 위아래로 병합

8) pd.concat([데이터프레임1, 데이터프레임2], axis=1) → 옆으로 병합

* key열값 == 10인 데이터를 찾고 싶다면?

  dataset[dataset.key == 10] #인덱스 조회라 부르고, 이는 매우 편리하다.

Pandas

범주형(Categorical) 변수 다루기

1) dataset = pd.Categorical(['low', 'high', 'medium', 'high', 'low'])

2) dataset.categories → 범주의 종류 조회 : 3개 값

3) dataset.get_values() → 범주 값 조회 : 5개 값

4) dataset.codes() → 범주의 코드 조회 : 5개 값 [1, 0, 2, 2, 0]

Pandas

범주형(Categorical) 변수 다루기

5) dataset.add_categories() → 범주 추가

6) dataset.remove_categories() → 범주 삭제

7) dataset.set_categories() → 범주 세팅 (추가/삭제 동시 수행)

8) dataset.value_counts() → 범주별 개수 세기

Pandas

연산

1) dataset 에 사칙연산 → value 값을 변경

2) dataset[열].count() → 해당 열의 값별 개수 세기

3) dataset[열].unique() → 해당 열의 고유값 목록 조회(NaN값 포함)

4) dataset[열].nunique() → 해당 열의 고유값 목록 조회(NaN값 미포함)

5) dataset.groupby(기준열) → 기준열 기준으로 그룹화

Pandas

데이터프레임 전체 약식 조회

1) dataset.describe() → dataset의 통계치 조회(numerical 데이터만)

2) dataset.describe(include='O') → dataset의 통계치 조회(categorical만)

3) dataset.describe(include='all') → dataset의 통계치 조회(둘다)

4) dataset.info() → nonnull값 개수, type, 데이터 크기 조회

Pandas

CSV 외 여러 종류 데이터 읽기

1) 엑셀 파일 읽기 → pd.read_excel()

2) json 파일 읽기 → pd.read_json()

3) html 파일 읽기 → pd.read_html()

4) sql 파일 읽기 → pd.read_sql()

저장은 read_x()를 to_x()로 바꾸면 된다.

Pandas

CSV 외 여러 종류 데이터 읽기 - HDF5

HDF5는 대용량 데이터를 관리하기 위한 라이브러리이자, 파일 포맷이다.

HDFStore를 이용한다. (내부적으로 Pytables 라이브러리를 사용하므로, 설치가 되어있어야한다.)

```
store = pd.HDFStore('data/store.h5') #create hdf5 store
store['dataset'] = dataset

store = pd.HDFStore('data/store.h5') #read hdf5 store
dataset = store['dataset']
```

Pandas

데이터 정제

1) 결측치 몇개나 있을까? → dataset.isnull().sum()

2) 결측치 버리기 → dataset.dropna()

* 결측치만 있는 행 버리기 → dataset.dropna(how='all')

* 결측치만 있는 열 버리기 → dataset.dropna(how='all', axis=1)

* 결측치 있는 열 버리기 → dataset.dropna(how='any')

Pandas

데이터 정제

3) 결측치 채우기 → dataset.fillna(채울값)

4) 해당 열 결측치 채우기 → (위의 값으로) → dataset[열].fillna(method='ffill')

5) 해당 열 결측치 채우기 → (아래 값으로) → dataset[열].fillna(method='bfill')

6) 해당 열 결측치 채우기 → (보간법으로) → dataset[열].interpolate()

* 결측치 외의 값을 바꾸고 싶다면? → replace({기존값 : 바꿀값}) / apply(lambda)

14 Matplotlib / Seaborn

Matplotlib / Seaborn

Matplotlib란? matplotlib

Matplotlib 패키지는 파이썬에서 자료를 차트나 플롯으로 시각화할 때 사용하는 대표적인 시각화 패키지다. 여러 차트와 플롯을 지원하며, 부가적인 기능들도 다양하게 지원하는 것이 Matplotlib의 강점이다.

Seaborn이란? Seaborn

Seaborn 패키지는 Matplotlib보다 다양한 색상과 스타일을 보강한 세련된 시각화 패키지다. 타일 지정을 위한 색상 팔렛트 기능, 커널 밀도, 다차원 복합 분포 등 다양한 기능을 지원하는 것이 강점이다.

Matplotlib.pyplot

1) 선 그래프 → plt.plot(x, y)

2) 산점도 그래프 → plt.scatter(x, y)

3) bar 그래프 → plt.bar(x, y)

4) barh 그래프 → plt.barh(x, y)

5) histogram → plt.histogram(x, bins=)

6) pie 그래프 → plt.pie(percent, label, autopct='%.1f%%')

Matplotlib.pyplot

그래프 부가 기능

1) 제목 → plt.title()

2) 축 이름 → plt.xlabel() / plt.ylabel()

3) 축 눈금 → plt.xticks() / plt.yticks()

4) 격자 → plt.grid()

5) 범례 → plt.legend()

Matplotlib.pyplot

그래프 부가 기능

6) 크기 → plt.figure(figsize=)

7) 서브 plot → fig = plt.figure()

fig.subplot()

8) plot 저장 → plt.savefig()

seaborn

1) heatmap → sns.heatmap(x, annot=True, annot_kws={size: 0.8})

2) bar → sns.factorplot(x, y, data, kind='bar')

3) box → sns.factorplot(x, y, data, kind='box')

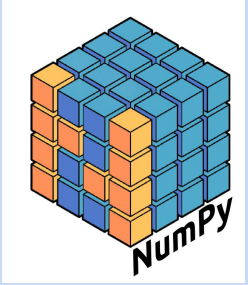
4) violin → sns.factorplot(x, y, data, kind='violin')

5) strip → sns.factorplot(x, y, data, kind='strip')

6) swarm → sns.factorplot(x, y, data, kind='swarm')

Numpy

Numpy



Numpy란?

Numpy 패키지는 수치해석용 패키지다. 파이썬은 기본적으로 배열 자료구조를 제공하지 않는데, 이를 Numpy에서 ndarray라는 배열자료구조를 제공한다. 이는 선형대수학에서 다루는 벡터와 행렬처럼 선형대수계산을 지원하는 자료구조를 의미한다.

numpy를 이요한 배열 연산은 상대적으로 매우 빠르고, 벡터화 연산은 리스트와 비교했을 때 혁신과도 같은 편리함을 제공해준다. 또한, 질의 기능을 지원하는 배열 인덱싱도 매우 편리하다. 이외의 리스트와 다음 2가지의 차이점도 기억해두자.

1) 모든 원소가 같은 자료형이어야 한다.
2) 원소의 갯수를 바꿀 수 없다.

Numpy

배열 생성

1) 0행렬 → np.zeros(행렬 shape)

2) 1행렬 → np.ones(행렬 shape)

3) 빈 행렬 → np.empty(행렬 shape)

4) 기존 데이터(data)와 같은 shape의 0행렬 → np.zeros_like(data)

5) 기존 데이터(data)와 같은 shape의 1행렬 → np.ones_like(data)

Numpy

배열 생성

6) 1부터 10까지 1씩 증가하는 원소 10개를 가진 행렬 → np.arange(1, 11, 1)

7) 1부터 10까지 동일한 간격으로 10개 원소를 가진 행렬 → np.linspace(1, 10, 10)

8) 10부터 10**10까지 log scale 간격으로 10개 원소를 가진 행렬 → np.logspace(1, 10, 10)

9) 난수를 가진 행렬(난수행렬) → (0~1사이) np.random.rand(행렬 shape)

→ (기대값=0, 표준편차=1 정규분포) np.random.randn(행렬 shape)

→ (정수) np.random.randint(범위시작점, 범위끝점, 행렬 shape)

Numpy

배열 조회

1) 배열의 shape → 배열.shape

2) 배열의 차원 → 배열.ndim

3) 배열의 type → 배열.dtype

4) 배열 인덱싱 → (3행 2열) 배열[3, 2]

→ (3의 배수인 원소들) 배열[배열%3==0]

Numpy

통계

1) 최대 / 최소 → np.max / np.min
2) argmax / argmin → np.argmax / np.argmin
3) 합 / 평균 / 중앙값 → np.sum / np.mean / np.median
4) 표준편차 / 분산 → np.std / np.var

Sklearn

Sklearn

Sklearn이란?

Sklearn 패키지는 머신러닝 패키지다. Sklearn은 튜토리얼 데이터셋 제공 / 데이터 전처리 / 다양한 머신러닝 모델 - 학습/검증 에 걸쳐 머신러닝에 관한 폭 넓은 기능을 제공하는 머신러닝 Generalist라고 볼 수 있다.

Sklearn의 가장 큰 강점은 매우 다양한 머신러닝 모델을 제공한다는 점이다. 단 딥러닝에 한해서는 기본적인 다층 퍼셉트론까지만 제공한다. 그래서, 엄밀히 말하자면 딥러닝을 제외한 머신러닝 Generalist 라고 부를 수 있다.

Sklearn

Sklearn 으로 머신러닝하기

1) 모델 불러오기 → from sklearn.모델클래스 import 모델

2) 모델 선언 → 변수 = 모델(중요 파라미터값 세팅)

3) 모델 학습 → 변수.fit(데이터) #지도학습 = x, y / 비지도학습 = x

4) 모델 검증 → 변수.score(데이터)

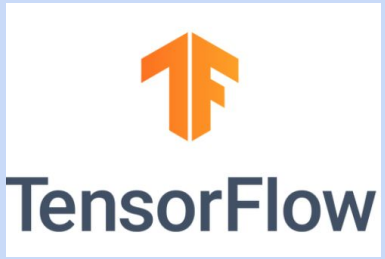
5) 모델 검증 결과 조회 → 변수.coef_ , 변수.intercepts_

6) 모델 예측 → 변수.predict() / 변수.predictproba()

Tensorflow

Tensorflow

Tensorflow란?



Tensorflow 패키지는 구글에서 만든 딥러닝을 위한 패키지다. Tensorflow 2 버전부터는 완벽하게 Keras 를 내장하고 있어, 코드의 편의성도 높였다.

파이토치, 카페 등 다른 패키지도 있지만 텐서플로가 가장 인기가 높다. 사이킷런이 딥러닝을 제외한 머신러닝 Generalist라면 , 텐서플로는 딥러닝 Specialist라고 부를 수 있다.

Tensorflow

tensorflow.keras 로 머신러닝하기

1) 모델 선언 → 변수 = models.sequence()

2) 모델 구축(층 쌓기) → 변수.add(layers.층종류())

3) 손실함수/최적화함수/평가지표 선택 → 변수.compile(loss, optimizer, metric)

4) 모델 학습 → 변수.fit(데이터) / 변수.train_on_batch(데이터)

5) 모델 검증 → 변수.evaluate(데이터)

Thank you

이원재(알디노)

ardino-lab.com

Ondslee0808@gmail.com

010-3501-4112

언제든 문의/상담 연락 주세요