



앱 개발 프로젝트

이가연

Table of Contents

- 반복
- Generate
- 서식

반복

DATE OF BIRTH:

27 ▾ January ▾ 1981 ▾

✓ January
February
March
April
May
June
July



이렇게 길게 나열해야 할 때

1. 여러 번 쓰기

```
const Column(  
  children: [  
    Text("January"),  
    Text("February"),  
    Text("March"),  
    Text("April"),  
    Text("May"),  
    Text("June"),  
    Text("July"),  
    Text("August"),  
    Text("September"),  
    Text("October"),  
    Text("November"),  
    Text("December"),  
  ],  
), // Column
```

2. List선언 후 generate 사용

```
List<String> months = [  
  "January",  
  "February",  
  "March",  
  "April",  
  "May",  
  "June",  
  "July",  
  "August",  
  "September",  
  "October",  
  "November",  
  "December"  
];
```

```
Column(  
  children: List.generate(  
    months.length,  
    (index) {  
      return Text(months[index]);  
    },  
  ), // List.generate  
) // Column
```

generate

반복해서 리스트를 만들 때 사용

파이썬의 list comprehension(리스트 컴프리헨션)과 유사

```
list1 = [i for i in range(6)]  
# [0, 1, 2, 3, 4, 5]
```

파이썬의 리스트 컴프리헨션

List(**[]**←이거)가 들어가는 곳에는 어디든 가능!

children: **[Text("Jan"), Text("Feb")]**와 같이 표현했으니 **generate**로 대체 가능

```
List.generate(개수, (index) {return 위젯(리스트[index])});
```

```
List.generate(개수, (index) => 위젯(리스트[index]));
```

=>는 한 줄짜리 (return만 있는) 함수 블록 선언 시 사용 가능

```
Column(  
  children: List.generate(  
    months.length,  
    (index) {  
      return Text(months[index]);  
    },  
  ), // List.generate  
, // Column
```

generate

```
final List<List<String>> secondThirdGrade = [
    ["교시", "시간", " ", "문항수", "과목"],
    ["1", "08:30 ~ 09:50", "80분", "45", "국어"],
    ["2", "10:10 ~ 11:50", "100분", "30", "수학"],
    ["3", "13:00 ~ 14:13", "70분", "45", "영어"],
    ["4", "14:30 ~ 15:00", "30분", "20", "한국사"],
    ["4", "15:08 ~ 15:38", "30분", "20", "탐구1선택"],
    ["4", "15:40 ~ 16:10", "30분", "20", "탐구2선택"],
];

final List<List<String>> firstGrade = [
    ["교시", "시간", " ", "문항수", "과목"],
    ["1", "08:30 ~ 09:50", "80분", "45", "국어"],
    ["2", "10:10 ~ 11:50", "100분", "30", "수학"],
    ["3", "13:00 ~ 14:13", "70분", "45", "영어"],
    ["4", "14:30 ~ 15:00", "30분", "20", "한국사"],
    ["4", "15:08 ~ 15:48", "40분", "20", "통합사회"],
    ["4", "15:50 ~ 16:20", "40분", "20", "통합과학"],
];

final List<List<String>> regularExam = [
    ["교시", "시간", " "],
    ["1", "08:50 ~ 09:40", "50분"],
    ["2", "10:00 ~ 10:50", "50분"],
    ["3", "11:10 ~ 12:00", "50분"],
];
```

시간표 데이터

- 2차원 배열

→ 2중 for문과 유사하게 generate도 2개 사용

교시	시간	
1	08:50 ~ 09:40	50분
2	10:00 ~ 10:50	50분
3	11:10 ~ 12:00	50분

시간표는 표이므로 Table 위젯 사용

→ TableRow 위젯들을 children에 넣어야 함

→ TableRow안에는 TableCell

➔ List.generate을 어떻게 구성해야 할까?

코드

```
Table(  
  children: List.generate(  
    시간표 행 개수,  
    (rowIndex) => TableRow(  
      children: List.generate(  
        시간표 열 개수,  
        (colIndex) => TableCell(  
          child: Text(각 칸에 들어갈 내용)  
        ),  
      ),  
    ),  
  ),  
)
```

서식

첫번째 행에만 색을 넣고 싶으면? →

현재 시간에 해당하는 행을 표시하려면? →

교시	시간	
1	08:50 ~ 09:40	50분
2	10:00 ~ 10:50	50분
3	11:10 ~ 12:00	50분

→ If문 활용 or 3항 연산자 사용

조건 ? 참일때 : 거짓일때

(if문은 TableRow를 따로 정의하기 때문에


행의 색 뿐만 아니라 글자 크기와 같은 다른 서식을 변경하기 용이,

하지만 코드가 길어지므로 자료에선 3항 연산자 기준으로 설명)

색깔 넣는 법

TableRow에 decoration 속성 추가,
BoxDecoration의 color 속성으로 색칠

여기에 3항 연산자 조건, true일
때는 BoxDecoration, false면 null



```
TableRow(  
  decoration: BoxDecoration(  
    color: Colors.grey,  
  ),  
  children: List.generate(  
    // 각 행에 들어갈 내용  
  ),  
)
```

서식

첫번째 행에만 색을 넣고 싶으면? →

현재 시간에 해당하는 행을 표시하려면? →

교시	시간	
1	08:50 ~ 09:40	50분
2	10:00 ~ 10:50	50분
3	11:10 ~ 12:00	50분

➔ 현재 시간을 받아오고, 표의 시간과 비교해
조건에 따라서 색을 칠하자!

해당 행의 시간과 현재 시간이 일치하는지 검사하는 1)함수를 선언하고,
2)조건문 안에 넣어 함수의 리턴값이 True일 때 색을 칠하도록 구성하자

3항 연산자 구성 예시

```
decoration: index == 0 ? BoxDecoration(color: Colors.grey) :  
            isCurrentTime(currentTimeTable[index][1]) ? BoxDecoration(color: Colors.red) : null
```

If문 구조로 나타내보면

```
if index == 0 :  
    # 회색  
else:  
    if isCurrentTime():  
        # 빨간색  
    else:  
        null
```


서식

해당 행의 시간과 현재 시간이 일치하는지 검사하는 함수는 어떻게 만들까?

Widget build함수 밖에, _MyHomePageState 클래스 안에 아래와 같은 함수를 선언

```
bool isCurrentTime(String timeString) { // 표의 시간 부분을 timeString으로 받아옴
    final now = DateTime.now(); // 현재 시간을 DateTime으로 받아옴
    // timeString을 split, indexing해서 DateTime형식으로 변환
    // isBefore, isAfter를 활용해 표의 시작 시간과 끝 시간 사이에 현재 시간이 있는지
    // 확인 후 true, false값 리턴
}
```

주석 부분을 잘 채워서 함수를 완성해봅시다!

서식

열의 너비를 조절하고 싶다면?

Table 위젯의 `columnWidths` 속성을 추가해 `<int, TableColumnWidth>` 맵 구조를 정의해준다
Python의 dictionary와 같음: `{0: ~~, 1: ~~}`

TableColumnWidth로 가능한 것들
(클릭 시 docs로 이동)

- FixedColumnWidth
- FlexColumnWidth
- FractionColumnWidth
- IntrinsicColumnWidth
- MaxColumnWidth
- MinColumnWidth

```
Table(  
    columnWidths: <int, TableColumnWidth>{  
        0: // 첫번째 열의 너비  
        1: // 두번째  
        2: // 세번째  
        // 필요한 만큼 선언  
    }  
  
    // 나머지 코드  
)
```

**앱에 추가하고 싶은 기능을
생각해보아요**

끝!