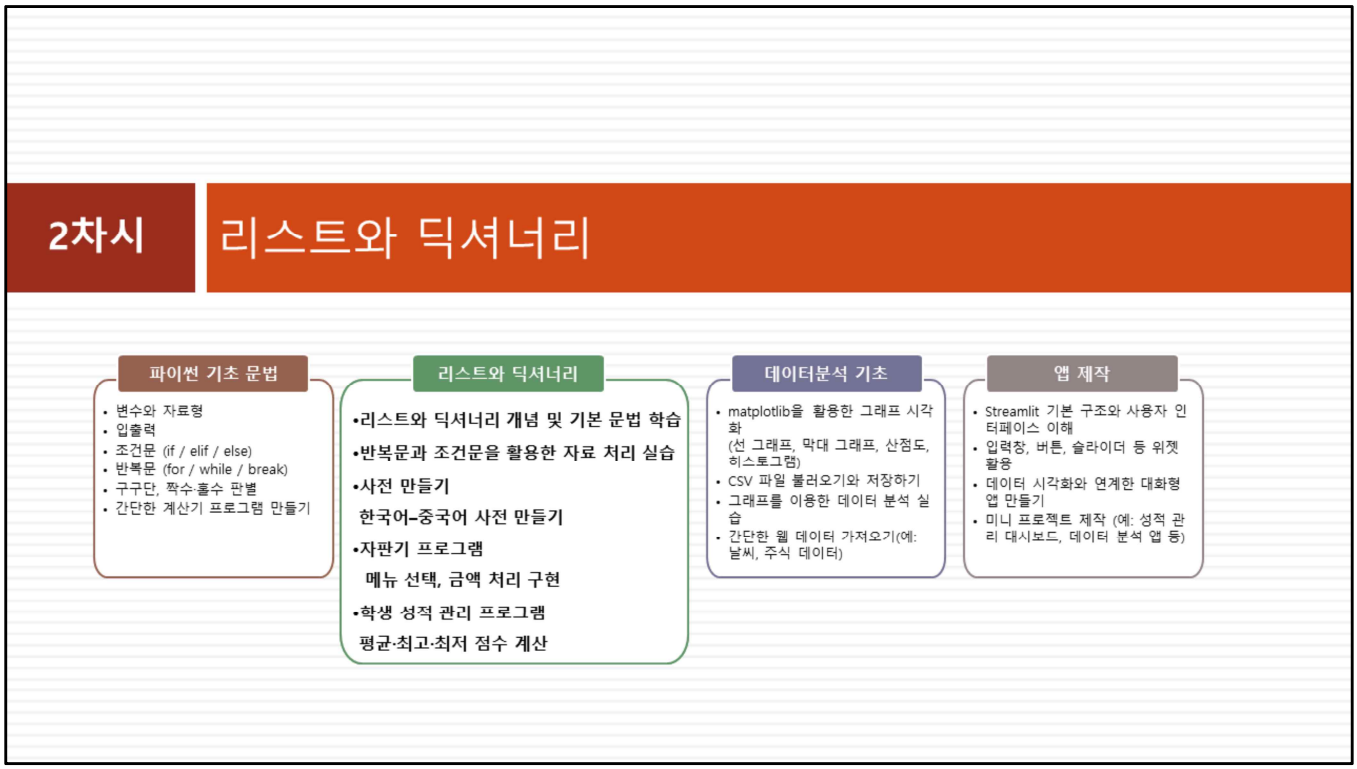


参考

- 리스트와 딕셔너리 개념 및 기본 문법 학습
  学习列表和字典的概念及基本语法
- 반복문과 조건문을 활용한 자료 처리 실습
  利用循环和条件语句进行数据处理练习
- 사전 만들기 → 한국어–중국어 사전 만들기
  制作词典 → 韩语–汉语词典
- 자판기 프로그램 → 메뉴 선택, 금액 처리 구현
  自动售货机程序 → 菜单选择与金额处理实现
- 학생 성적 관리 프로그램 → 평균·최고·최저 점수 계산
  学生成绩管理程序 → 计算平均分、最高分、最低分

List(리스트)와 인덱스

20

- list는 여러 개의 데이터를 저장하는 공간으로, 각 데이터를 항목(item) 또는 요소(element)라고 부름
  - years = [2023,2024,2025,2026]
  - score = [90,80,10,87,75]
  - fruits =['apple', 'banana', 'cherry']
- 리스트의 각 항목은 인덱스(index)로 표현하며 0부터 시작함
  - score[0], score[1]...score[4]

CODE

```
score= [90,80,20,87,75]
print(score)
print(score[0])
print(score[4])

for a in score:
  print(a)
```

参考

- 列表（list）是用来存储多个数据的空间，每个数据称为项（item）或元素（element）
- 列表中的每个项都用索引（index）表示，并且从 0 开始。

리스트 다루기

21

- 리스트에 값 추가 : append() 함수
  - 리스트이름.append(값)
  - value1.append(100)
- 음수 인덱싱 : 마지막 항목부터 인덱스 -1, -2...
  - print(value1[-1])
- 리스트의 길이 : len() 함수
  - print(len(value1))
- 리스트 슬라이싱 : [시작:종료]를 이용하여 일부 항목 추출
  - print(value1[1:4]) -> 인덱스1부터 [인덱스4]-1인 인덱스 3번까지 추출
    - [1, 2, 3]
  - print(value1[:4]) -> 시작 또는 종료 인덱스를 생략하면 처음부터 끝까지
    - [0, 1, 2, 3]
  - print(value1[-5:]) -> -5번지부터 끝까지 추출
    - [5, 6, 7, 8, 9]

CODE

```
value1 = []
for i in range(0,10):
  value1.append(i)
print(value1)

value2 = []
for i in range(1,11):
  value2.append(i*i)
print(value2)
```

```
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

参考

操作列表

- 添加：append() 函数
  - 列表名.append(值)
  - value1.append(100)
- 负索引：从最后一个元素开始，索引为 -1, -2...
  - print(value[-1])
- 列表的长度：len() 函数
  - print(len(value1))
- 列表切片：使用 [开始:结束] 提取部分元素
  - print(value1[1:4]) → 从索引1到索引3 [1,2,3]
  - print(value1[:4]) → 省略开始索引时，从开头到结束 [0,1,2,3]
  - print(value1[-5:]) → 从倒数第5个到最后 [5,6,7,8,9]

반복문으로 리스트에 랜덤 값 추가하기

22

1. 빈 리스트 생성
2. 1부터 100 사이의 랜덤 숫자 10개 추가

CODE

```
import random
numbers = []
for i in range(10):
    value = random.randint(1, 100)
    numbers.append(value)
print("생성된 리스트:", numbers)
```

参考

```
import random  # 랜덤 모듈 불러오기 / 导入随机模块

# 빈 리스트 생성 / 创建一个空列表
numbers = []

# 1부터 100 사이의 랜덤 숫자 10개 추가 / 添加 10 个 1~100 之间的随机整数
for i in range(10):
    value = random.randint(1, 100)  # 1~100 사이의 정수 생성 / 生成 1~100 之间的整数
    numbers.append(value)           # 리스트에 추가 / 添加到列表中

print("생성된 리스트:", numbers)  # 생성된 리스트 출력 / 输出生成的列表
```

리스트 항목의 최대, 최소, 합 구하기

23

- max(리스트) : 항목 중 최댓값
- min(리스트) : 항목 중 최솟값
- sum(리스트) : 항목들의 합
- 리스트.sort() : 정렬

CODE

```
import random
students = ["민지", "철수", "지영", "현우", "수빈"]
students.sort()
print(students)
winner = random.choice(students)
print("오늘의 발표자는:", winner)
```

CODE

```
scores= [90,80,70,87,65]
print('MAX:', max(scores))
print('MIN:', min(scores))
print('SUM:', sum(scores))
print('AVG:', sum(scores)/len(scores))
```

```
MAX: 90
MIN: 65
SUM: 392
AVG: 78.4
```

参考

获取列表元素的最大值、最小值与总和

- max(列表)：元素中的最大值
- min(列表)：元素中的最小值
- sum(列表)：元素的总和
- 列表.sort()：排序

random.choice() 함수 이용하여 이름 만들기

24

- random.choice(리스트)
  - 리스트(또는 튜플, 문자열 등 시퀀스 자료형) 안에서 **무작위로 하나의 요소를 선택**해서 반환 함
  - import random
  - names = ["철수", "영희", "민수", "지은"]
  - print(random.choice(names))

CODE

```
import random

# 성(姓) / 姓氏
last_names = ["김", "이", "박", "최", "정"]

# 이름(名) 앞 음절 / 名字第一音节
first_parts = ["정", "채", "지", "수", "준", "하", "서", "영"]

# 이름(名) 뒷 음절 / 名字第二音节
second_parts = ["호", "우", "현", "빈", "원", "연", "아"]

# 무작위 이름 생성 / 随机生成名字
for i in range(5):
    name = random.choice(last_names) +
    random.choice(first_parts) +
    random.choice(second_parts)
    print("i+1, ":", name)
```

```
1 : 김서빈
2 : 정하빈
3 : 박서우
4 : 이채호
5 : 박하빈
```

参考

• **random.choice()：**从列表（或元组、字符串等）中 **随机选择一个元素** 并返回。

```
import random
# 중국 성씨 (常见姓氏)
last_names = ["王", "李", "张", "刘", "陈", "杨", "赵", "黄", "周", "吴"]
# 이름 앞 글자 (名字第一字)
first_parts = ["伟", "芳", "娜", "敏", "静", "秀", "强", "磊", "军", "洋"]
# 이름 뒷 글자 (名字第二字，可选)
second_parts = ["华", "明", "红", "丽", "杰", "刚", "峰", "玲", "龙", "超", "玉", "涛"]

# 무작위 이름 생성 (1글자 또는 2글자 이름) / 随机生成名字（一个字或两个字）
for i in range(5):
    last = random.choice(last_names)
    first = random.choice(first_parts)
    # 50% 확률로 이름을 한 글자 또는 두 글자 생성 / 50% 概率生成一个字或两个字
    if random.choice([True, False]):
        name = last + first
    else:
        name = last + first + random.choice(second_parts)
    print("随机中文名字:", name)
```

딕셔너리(dictionary)

25

- 하나 이상의 데이터를 저장하는 자료이며, 키(key)와 값(value)의 쌍으로 이루어짐. 값에 접근할 때 키를 이용
- 생성 : 딕셔너리이름 = {키1:값1, 키2:값2,...}
  - scores = {"kim": 85, "lee": 92, "choi": 78 }
- 추가 또는 변경 : 딕셔너리이름['키'] = 값
  - scores["park"] = 88
  - scores["kim"] = 90
- 키(key)는 고유한 값으로 중복값 허용 안됨
- 주요 함수
  - dict.keys() : 키 모음
  - dict.values() : 값 모음
  - dict.items() : (키, 값) 쌍 모음
  - dict.pop(key) : 키 항목 삭제

CODE

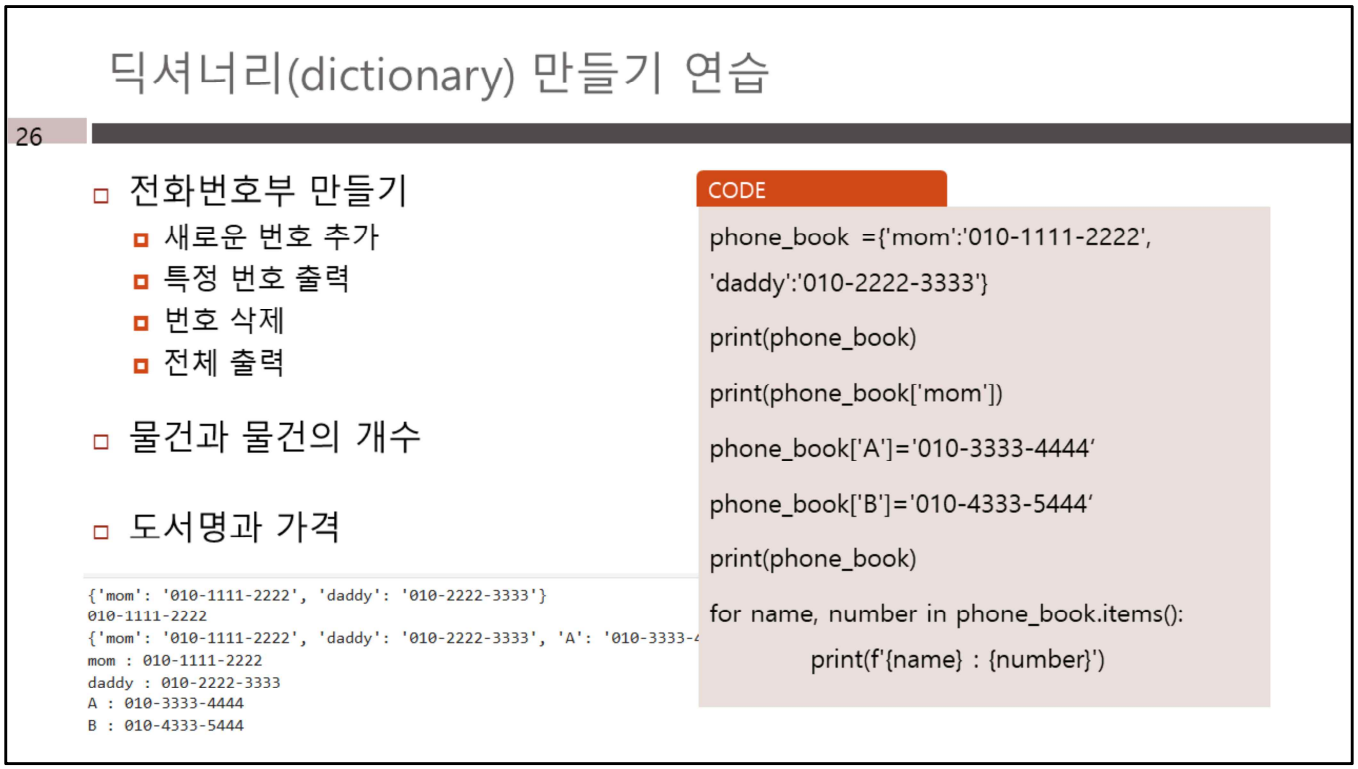
```
# 학생 이름-점수 딕셔너리 만들기
scores = {"kim": 85, "lee": 92, "choi": 78 }
# 특정 학생 점수 출력
print("lee:", scores["lee"])
scores["park"] = 88 #새로운 학생 추가
scores["kim " ] = 90  #기본 학생 점수 변경
# 전체 출력
for name, score in scores.items():
    print(f"{name} : {score}점")
# 평균 점수
print("평균 점수:", sum(scores.values()) / len(scores))
```

```
lee : 92
kim : 90점
lee : 92점
choi : 78점
park : 88점
평균 점수: 87.0
```

参考

字典（dictionary）

- 一种可以存储一个或多个数据的结构，由键（key）和值（value）的对组成。访问数据时通过键来获取。
- 创建：字典名 = {键1:值1, 键2:值2, ...}
- 例:
  - ✓ scores = {"kim":85, "lee":92, "choi":78}
- 添加或修改：字典名['键'] = 值
- 例:
  - ✓ scores["park"] = 88
  - ✓ scores["kim"] = 90
- 键（key）必须唯一，不允许重复
- 主要函数
  - ✓ dict.keys()：键集合
  - ✓ dict.values()：值集合
  - ✓ dict.items()：(键, 值) 对集合
  - ✓ dict.pop(key)：删除指定键的项



参考

딕셔너리 만들기 연습 (字典练习)

- **电话簿 (전화번호부 만들기)**
  - ✓添加新号码 (새로운 번호 추가)
  - ✓输出特定号码 (특정 번호 출력)
  - ✓删除号码 (번호 삭제)
  - ✓输出全部 (전체 출력)
- **物品与数量 (물건과 물건的 개수)**
- **书名与价格 (도서명과 가격)**

딕셔너리 기능 이용하여 단어사전 만들기

27

```
dictionary = {
    "assignment": "과제",    "exam": "시험",    "professor": "교수님",
    "major": "전공",    "credit": "학점",    "library": "도서관",
    "presentation": "발표",    "project": "프로젝트",    "club": "동아리",
    "internship": "인턴십"
}
while True:
    print("₩n=== Simple English-Korean Dictionary ===")
    print("1) 영어 → 한국어 조회 (English → Korean Search)")
    print("2) 단어 추가 (Add Word)")
    print("3) 단어 삭제 (Delete Word)")
    print("4) 전체 단어 보기 (Show All Words)")
    print("0) 종료 (Exit)")
    choice = input("메뉴 선택 (Choose a menu): ")
    if choice == "1":
        word = input("Enter an English word: ")
        if word in dictionary:
            print("Korean:", dictionary[word])
        else:
            print("Word not found in dictionary.")
    elif choice == "2":
        en = input("English word to add: ")
        ko = input("Korean meaning: ")
        dictionary[en] = ko
        print("Word added!")
    elif choice == "3":
        en = input("English word to delete: ")
        if en in dictionary:
            del dictionary[en]
            print("Word deleted!")
        else:
            print("Word not found in dictionary.")
    elif choice == "4":
        print("=== All Words ===")
        for k, v in dictionary.items():
            print(k, ":", v)
    elif choice == "0":
        print("Program ended.")
        break
    else:
        print("Please enter a valid menu number.")
```

```
=== Simple English-Korean Dictionary ===
1) 영어 → 한국어 조회 (English → Korean Search)
2) 단어 추가 (Add Word)
3) 단어 삭제 (Delete Word)
4) 전체 단어 보기 (Show All Words)
0) 종료 (Exit)
메뉴 선택 (Choose a menu):  2
English word to add:  cafeteria
Korean meaning:  카페테리아
Word added!

=== Simple English-Korean Dictionary ===
1) 영어 → 한국어 조회 (English → Korean Search)
2) 단어 추가 (Add Word)
3) 단어 삭제 (Delete Word)
4) 전체 단어 보기 (Show All Words)
0) 종료 (Exit)
메뉴 선택 (Choose a menu):  4
=== All Words ===
assignment : 과제
exam : 시험
professor : 교수님
major : 전공
credit : 학점
library : 도서관
presentation : 발표
project : 프로젝트
club : 동아리
internship : 인턴십
cafeteria : 카페테리아
```

参考

딕셔너리 기능을 이용하여 단어 사전 만들기(利用字典功能制作词典)

1. **초기 단어 저장 (保存初始词汇)**
   - ✓프로그램 시작 시 한국어–중국어 단어쌍 저장(程序一开始保存一些韩语–汉语单词对)
2. **메뉴 출력 (显示菜单)**
   - ✓실행 후 메뉴를 출력하고, 사용자가 기능을 선택(运行后会显示菜单，用户可以选择需要的功能)
3. **주요 기능 (主要功能)**
   - ✓단어 조회 → 한국어 단어를 입력하면 대응되는 중국어 출력(查询单词 → 输入韩语单词后，显示对应的中文意思)
   - ✓단어 추가 → 새로운 한국어–중국어 단어를 추가 가능(添加单词 → 可以向词典中添加新的韩语和中文单词)
   - ✓단어 삭제 → 해당 단어가 있으면 삭제(删除单词 → 如果输入的韩语单词存在，就从词典中删除)
   - ✓전체 보기 → 저장된 모든 단어쌍 출력(查看全部单词 → 可以查看词典中保存的所有单词)
   - ✓종료 → 프로그램을 종료(退出 → 结束程序)
4. **확장 아이디어 (扩展思路)**
   - ✓단어만 교체하면 다양한 주제의 사전 제작 가능(只要更换词汇，就能制作出各种主题的词典)

```
dictionary = {
    "作业": "과제",
    "考试": "시험",
    "教授": "교수님",
    "专业": "전공",
    "学分": "학점",
```

```
    "图书馆": "도서관",
    "发表": "발표",
    "项目": "프로젝트",
    "社团": "동아리",
    "实习": "인턴십"
}
```

Vending Machine

28

```
# Vending machine menu (dictionary: {item: price})
menu = {
    "Coke": 1200,
    "Sprite": 1000,
    "Coffee": 1500,
    "Water": 800
}
print("=== Vending Machine ===")
print("Menu and Prices:")
for item, price in menu.items():
    print(f"{item}: {price} won")
while True:
    choice = input("₩nSelect a drink (type 'q' to quit): ")
    if choice == "q":
        print("QUIT")
        break
    if choice not in menu:
        print("Please choose again.")
        continue
price = menu[choice]
print("Price:", price, "won")
money = int(input("Insert money: "))
if money >= price:
    change = money - price
    print(f"You got {choice}. Change: {change} won")
else:
    print("Not enough money. Please try again.")
```

```
=== Vending Machine ===
Menu and Prices:
Coke: 1200 won
Sprite: 1000 won
Coffee: 1500 won
Water: 800 won

Select a drink (type 'q' to quit): Coke
Price: 1200 won
Insert money: 2000
You got Coke. Change: 800 won

Select a drink (type 'q' to quit): q
QUIT
```

参考

프로그램 개요 (程序简介) :자동판매기 프로그램(自动售货机程序)

1. **메뉴 정의 (菜单定义)**
   - ✓딕셔너리에 음료 이름과 가격을 저장(用字典存储饮料名称和价格)
2. **사용자 입력 반복 (用户输入循环)**
   - ✓사용자가 음료를 선택하고, 'q' 입력 시 프로그램 종료(选择饮料，输入 'q' 时退出程序)
3. **기능 (功能)**
   - ✓선택한 음료의 가격을 보여줌(显示所选饮料的价格)
   - ✓금액을 입력받아 거스름돈을 계산하고 음료를 제공(投入金额后，计算找零并提供饮料)
   - ✓금액이 부족하면 안내 메시지를 출력(如果金额不足，提示用户)

성적 입출력 프로그램

29

```
scores = {"kim": 85, "lee": 92, "choi": 78}
# Initial scores
print("=== Initial Scores ===")
for name, score in scores.items():
    print(f"{name} : {score} points")
# Add new scores
print("₩n=== Add Scores ===")
while True:
    name = input("Enter student name to add (q to quit): ")
    if name == "q":
        break
    score = int(input(f"Enter {name}'s score: "))
    scores[name] = score
    print(f"{name}'s score has been added!")
# Show all scores
print("₩n=== All Scores ===")
for name, score in scores.items():
    print(f"{name} : {score} points")

# Average score
print("₩nAverage Score:", sum(scores.values()) / len(scores))

# Search scores
print("₩n=== Search Scores ===")
while True:
    search = input("Enter student name to search (q to quit): ")
    if search == "q":
        print("Search ended.")
        break
    if search in scores:
        print(f"{search}'s score is {scores[search]} points.")
    else:
        print("Student not found.")
```

```
=== Initial Scores ===
kim : 85 points
lee : 92 points
choi : 78 points

=== Add Scores ===
Enter student name to add (q to quit):  park
Enter park's score:  90
park's score has been added!
Enter student name to add (q to quit):  q

=== All Scores ===
kim : 85 points
lee : 92 points
choi : 78 points
park : 90 points

Average Score: 86.25

=== Search Scores ===
Enter student name to search (q to quit):  kim
kim's score is 85 points.
Enter student name to search (q to quit):  lee
lee's score is 92 points.
Enter student name to search (q to quit):  1
Student not found.
Enter student name to search (q to quit):  q
Search ended.
```

参考

- **학생들의 성적을 딕셔너리로 관리하는 프로그램(用字典来管理学生成绩的程序)**
  - ✓기능: 성적 추가, 전체 출력, 평균 계산, 성적 검색(功能: 添加成绩、输出全部、计算平均分、查询成绩)
- **실행 순서 (运行顺序)**
  1. **초기 성적 출력**
     - 기존 딕셔너리에 저장된 학생 이름과 성적을 출력(输出字典中已保存的学生姓名和成绩)
  2. **성적 추가**
     - 사용자로부터 학생 이름과 성적을 입력받아 딕셔너리에 추가(输入新的学生姓名和成绩，并添加到字典中)
     - q 입력 시 추가 종료(输入 q 结束添加)
  3. **전체 성적 출력**
     - 현재 저장된 모든 학생과 성적을 출력(输出所有已保存的学生成绩)
  4. **평균 점수 계산**
     - 모든 성적을 합산 후 인원수로 나누어 평균 계산(计算所有成绩的总和除以人数，得到平均分)
  5. **성적 검색**
     - 사용자로부터 이름을 입력받아 성적을 조회(输入学生姓名并查询成绩)
     - 존재하지 않으면 "Student not found." 출력(若不存在，输出 "未找到该学生")
     - q 입력 시 검색 종료(输入 q 结束查询)