Programming Language

컴퓨터개론

(Introduction to Computer Systems)

GEN1030

Python

(Recap) Python

- 최근 가장 활발히 사용되는 언어 중 하나
  - Python 2: 2020년 공식 지원 종료
  - Python 3: Python 2의 문제점 개선, 현재 표준
- 인터프리터(interpreter) 언어
  - 컴파일 과정 없이 작성된 코드 한 줄 씩 읽어가며 실행
- 타입 선언 없음
  - 실행 중 타입 결정 - **동적 타이핑(Dynamic Typing)**

(Recap) Python – 기본 문법

- 자료형(Data Type)
  - 변수에 저장되는 데이터의 종류

- 정수형(int)
  - 소수점이 없는 숫자

x = 10

- 실수형(float)
  - 소수점이 있는 숫자

x = 3.14

- 불리언형(bool)
  - 참과 거짓(True or False) 나타냄

flag = True

(Recap) Python – 기본 문법

- 자료형(Data Type)
  - 변수에 저장되는 데이터의 종류

- 문자열(str)
  - 문자(글자)들의 집합
  - 따옴표로 감싸면 문자로 인식 ('…' 혹은 "…")

```
name = "Kim"
name = 'Lee'
```

- 리스트(list)
  - 순서가 있는 여러 데이터의 목록
  - 대괄호 사용
  - 인덱스(index)로 데이터 접근

```
x = [1, 2, 3]
y = ["a", "b", "c"]
x[2] = 10
y[1] = "x"
```

- 세트(set)
  - 중복이 없는 데이터들의 집합
  - 중괄호 사용
  - 순서가 없음, index로 접근 불가

```
s = {3, 1, 2}
s.add(2)      → s = {3, 1, 2}
s.add(4)      → s = {3, 1, 2, 4}
s.remove(2)   → s = {3, 1, 4}
```

(Recap) Python – 기본 문법

- 자료형(Data Type)
  - 변수에 저장되는 데이터의 종류

- 딕셔너리(dict)
  - 키(Key)와 값(Value)의 쌍으로 데이터를 저장
  - 중괄호를 사용

    student_name_id = {"alice": 20260001, "bob": 20260003}

  - 키 사용해서 값에 접근

    student_name_id["alice"]

    student_name_id["bob"] = 20260002

    → {"alice": 20260001, "bob": **20260002**}

(Recap) Python – 기본 문법

- 입출력
  - 프로그램은 사용자와 입력/출력으로 상호작용

- 입력(input)
  - 사용자로부터 값을 입력 받음
  - 데이터를 받아 변수에 저장

```
name = input()
```

- 출력(output)
  - 결과를 보여줌
  - 변수에 담긴 값이나 결과를 화면에 보여줌

```
print("Hello")
```

```
name = input()
print(name)
```

(Recap) Python – 기본 문법

- 조건문(Conditional Statement)
  - 주어진 조건(Condition)이 참(True)인지 거짓(False)인지에 따라 프로그램 실행 흐름을 바꾸는 문법
  - if (elif) else 구조
  - Python에서는 **들여쓰기(indentation)** 강제

```
if x == 0:
    print("zero")
elif x > 0:
    print("positive")
else:
    print("negative")
```

(Recap) Python – 기본 문법

- 반복문(Loop Statement)
  - 동일하거나 유사한 작업을 정해진 횟수만큼, 혹은 특정 조건이 만족할 때까지 되풀이하는 문법
  - for, while

```
for i in range(3):
    print(i)
```

range(n): 0이상 n 미만 정수 범위 생성

→ 출력 0
1
2

```
x = 0
while x < 3:
    print("Hi")
    x = x +1
```

→ 출력 "Hi"
"Hi"
"Hi"

(Recap) Python – 기본 문법

- 함수(function)
  - 특정 작업을 수행하는 코드의 묶음
  - 필요할 때마다 재사용 가능
    - 같은 코드 반복하지 않을 수 있음
  - 요소
    - 입력(input, parameter): 함수가 작업 위해 외부에서 전달받는 데이터
    - 처리(process, body): 실제 실행 코드
    - 출력(output, return): 작업 끝내고 최종적으로 내놓는 결과물
  - "**def**" 키워드 사용

def 함수이름(*입력*):
    처리
    return *출력*

```
def add(a, b):
    result = a + b
    return result
```

(Recap) Python – 기본 문법

- 함수(function)
  - **호출(call)**되어야 실제 실행 됨

```
def add(a, b):
    result = a + b
    print(result)
    return result

x = add(2, 5)
print(x)
print(result)
```

→ “7”

→ “7”

→ ?

  - 일반적으로 함수 내에서 선언한 변수는 함수가 종료되면 사라짐

(Recap) Python – 기본 문법

- 라이브러리(library)
  - 특정 기능을 수행하기 위해 미리 작성 된 코드 묶음/집합체
  - 복잡한 기능을 직접 구현하지 않아도 됨
    - 재사용
  - Python은 다양한 라이브러리가 잘 구축되어 있음
  - "import" 키워드 사용해서 외부의 기능을 현재 코드 안으로 가져옴

```
import math
print(math.pi)
```

```
import random
x = random.randint(1, 45)
print(x)
```

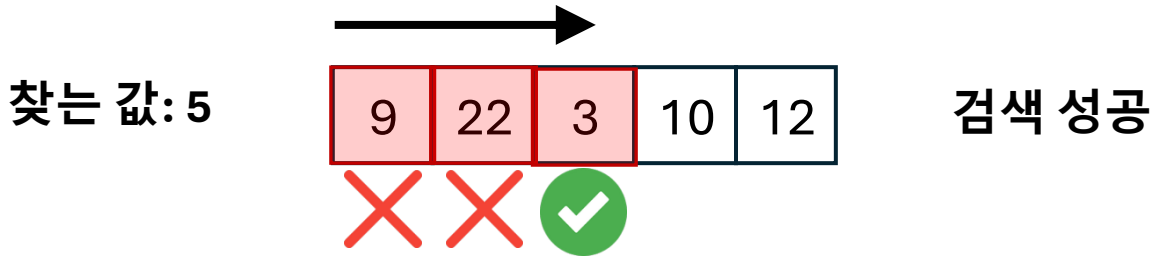
Python – Examples

- 반복문 + 함수

```
def sum_up_to_n(n)
        total = 0

        for i in range(1, n+1):
                total += i

        return total

result = sum_up_to_n(5)
print(result)

print(sum_up_to_n(4))
```

Python – Examples

- Sequential search
  - 일렬로 나열된 데이터를 처음부터 끝까지 순서대로 검색하는 방법




```
def seq_search(arr, target, len_arr):
    for i in range(len_arr):
        if arr[i] == target:
            return i
    return -1
```

```
nums = [9, 22, 3, 10, 12]
target = 3
len_arr = 5
idx = seq_search(nums, target, len_arr)
print(idx)
```

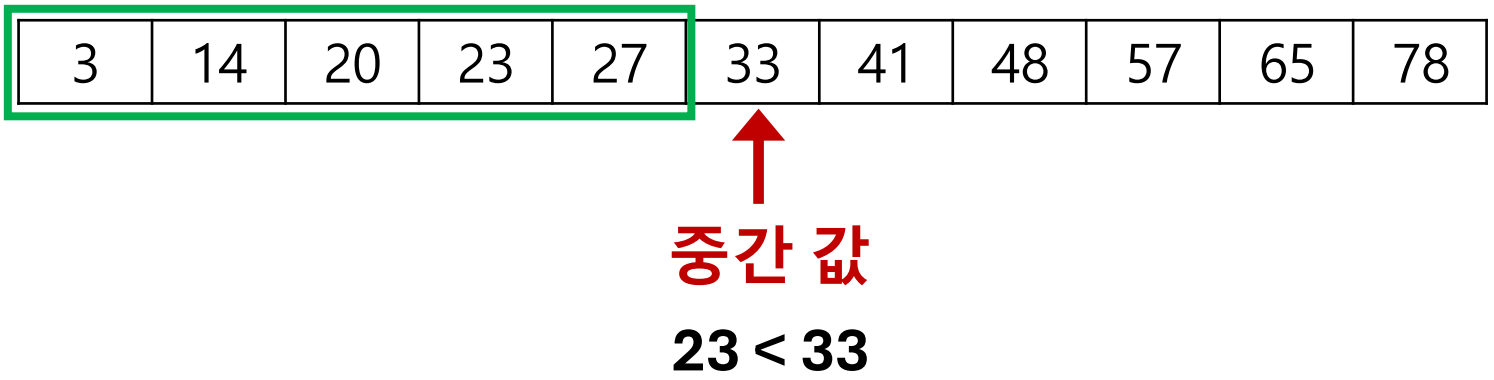
Python – Examples

- Binary search
  - 데이터 가운데에 위치한 항목을 검색 값과 비교
    - 검색 값이 더 크면 오른쪽 부분 검색
    - 검색 값이 더 작으면 왼쪽 부분 검색
  - **검색 대상인 데이터 개수를 평균적으로 ½씩 줄임**

이진 검색(Binary Search)

- Example

찾는 값: 23



이진 검색(Binary Search)

- Example

찾는 값: 23

이진 검색(Binary Search)

- Example

찾는 값: 23

중앙 위치?
- 홀수: 중앙
- 짝수: 왼쪽 공간에서 가장 큰 수

이진 검색(Binary Search)

- Example

찾는 값: 23

중앙 위치?
- 홀수: 중앙
- 짝수: 왼쪽 공간에서 가장 큰 수

Python – Examples

- Binary search

```
def binary_search(arr, target):
    low = 0
    high = len(arr) - 1

    while low <= high:
        mid = (low + high) // 2
        if arr[mid] == target:
            return mid
        elif arr[mid] > target:
            high = mid - 1
        else:
            low = mid + 1
    return -1
```

```
nums = [1, 2, 4, 5, 8]
target = 5

idx = binary_search(nums, target)
print(idx)
```

```
import bisect

nums = [1, 2, 4, 5, 8]
target = 5

idx = bisect.bisect_left(nums, target)
print(idx)
```

Summary

- Python
  - Data type
  - Condition / Loop statement
  - Function
  - Examples