

프로그래밍2 튜터링

(1회차)

조세연

Intro

- 조세연
- 숭실대학교 미디어경영학과 1학년
- 5년차 개발자 (C++)

튜터링 일정

- 매주 화요일 오후 9시 30분 부터
- 60분씩!
- 총 10회 (2022.10.18 ~ 2022.12.20) => 변경 가능
- 중간 및 기말고사 준비

목표

- 1학기에 배웠던 파이썬을 되짚어 보자!!
- 자료형
- 제어문
- 함수

Python?



- 파이썬(Python)은 1990년 암스테르담의 귀도 반 로섬이 개발한 인터프리터 언어.
- 인터프리터 언어?
 - 프로그래밍 언어의 소스 코드를 바로 실행하는 컴퓨터 프로그램 또는 환경
 - 원시 코드를 기계어로 번역하는 컴파일러와 대비되는 방식
- 파이썬의 사전적 의미는 고대 신화에 나오는 파르나소스 산의 동굴에 살던 큰 뱀을 뜻하며, 아폴로 신이 델파이에서 파이썬을 퇴치했다는 전설이 있음. -> 대부분의 파이썬 책 표지와 아이콘이 뱀 모양으로 그려져 있는 이유!!
- 컴퓨터 프로그래밍 교육을 위해 많이 사용되나, 기업의 실무에서도 많이 쓰임
- 공동 작업과 유지 보수가 매우 쉽고 편함

파이썬의 장점

- 파이썬은 인간다운 언어
- 문법이 쉬워 빠르게 배울 수 있음
- 무료지만 강력함
- 간결함
- 개발 속도가 빠름

자료형

숫자형

- 정수형: 정수형(Integer)이란 말 그대로 정수를 뜻하는 자료형
- 실수형: 실수형(Floating-point)은 소수점이 포함된 수를 뜻하는 자료형
- 8진수: 8진수(Octal)를 만들기 위해서는 숫자가 0o 또는 0O(숫자 0 + 알파벳o)로 시작
- 16진수: 16진수(Hexadecimal)를 만들기 위해서는 0x로 시작

자료형

문자열 - 1

- 문자열: 문자열(String)이란 문자, 단어 등으로 구성된 문자들의 집합
- 문자열 만드는 법
 - 큰 따옴표(“)로 양쪽 감싸기
 - 작은 따옴표(‘)로 양쪽 감싸기
 - 큰 따옴표 3개를 연속(“””)으로 써서 양쪽 감싸기
 - 작은 따옴표 3개를 연속(‘’)으로 써서 양쪽 감싸기

자료형

문자열 - 2

- 이스케이프 코드: 프로그래밍을 할 때 사용할 수 있도록 미리 정의해둔 “문자 조합”
 - `\n`: 문자열 안에서 줄을 바꿀 때 사용
 - `\t`: 문자열 사이에 탭 간격을 줄 때 사용
 - `\\`: 문자 `\`를 그대로 표현할 때 사용
 - `\'`: 작은따옴표(')를 그대로 표현할 때 사용
 - `\"`: 큰따옴표(")를 그대로 표현할 때 사용

자료형

리스트-1

- 리스트: 데이터를 묶어서 관리할 수 있는 자료형 중 하나
- 리스트 만드는 법
 - 리스트명 = [요소1, 요소2, 요소3, ...]
 - 리스트명 = [] (빈 리스트)
 - 리스트명 = list(요소1, 요소2, 요소3, ...)
 - 리스트명 = list() (빈 리스트)

자료형

리스트의 인덱싱

```
>>> a = [1, 2, 3]
```

```
>>> a
```

출력 결과: [1, 2, 3]

```
>>> a[0]
```

출력결과: 1

```
>>> a[0] + a[2]
```

출력결과: 4

```
>>> a[-1]
```

출력결과: 3

```
>>> a = [1, 2, 3, ['a', 'b', 'c']]
```

```
>>> a[-1]
```

출력결과: ?

```
>>> a[-1][0]
```

출력결과: ?

자료형

리스트의 슬라이싱

- 인덱싱 방법
 - [시작 idx : 끝 idx + 1]
 - [: 끝 idx + 1]
 - [시작 idx :]
 - [:]

```
>>> a = [1, 2, 3, 4, 5]
```

```
>>> a[0 : 2]
```

출력결과: [1, 2]

```
>>> b = a[ :2 ]
```

```
>>> c = a[ 2: ]
```

```
>>> b
```

출력결과: ?

자료형

튜플

- 리스트와의 차이점
 - 리스트는 [] 으로 둘러써지만, 튜플은 () 으로 둘러쌌
 - 리스트는 그 값의 생성, 삭제, 수정이 가능하지만 튜플은 값을 바꿀 수 없다.
-> 튜플의 값을 삭제하거나 변경하려 하면 에러가 발생!!

자료형

튜플 인덱싱 및 슬라이싱

인덱싱

```
>>> t1 = (1, 2, 'a', 'b')
```

```
>>> t1[0]
```

출력결과: 1

```
>>> t1[3]
```

출력결과: ?

슬라이싱

```
>>> t1 = (1, 2, 'a', 'b')
```

```
>>> t1[ 1: ]
```

출력결과: ?

* 리스트의 인덱싱 & 슬라이싱 방법과 동일

자료형

딕셔너리

- 단어 그대로 사전이라는 뜻으로, key-value로 구성되어 있음
- 리스트나 튜플처럼 순서적으로 해당 요솟값을 구하지 않고, key-value로 얻음
- 딕셔너리 만드는 법
 - {Key1: Value1, Key2: Value2, Key3: Value3, ...}
 - key는 변하지 않아야하며, value는 변할 수 있음

자료형

집합 자료형

- 집합 자료형은 파이썬 2.3부터 지원하기 시작한 자료형으로, 집합에 관련된 것을 쉽게 처리하기 위해 만든 자료형
- `set()` 키워드를 사용해 만들 수 있음
 - `s1 = set([1, 2, 3])`
 - `s2 = set("Hello") => 출력결과: {'e', 'H', 'l', 'o'}`
- 중복을 허용하지 않으며, 순서가 없음
- 인덱싱으로 접근하려면 리스트나 튜플로 변환 후 접근해야함

자료형

교집합, 합집합, 차집합

```
>>> s1 = set( [1, 2, 3, 4, 5, 6] )
```

```
>>> s1 = set( [ 4, 5, 6, 7, 8, 9] )
```

- 교집합

- $s1 \& s2$

- `s1.intersection(s2)`

- 합집합

- $s1 \mid s2$

- `s1.union(s2)`

- 차집합

- $s1 - s2$

- `s1.difference(s2)`

자료형

불

- 불(bool) 자료형이란 참(True)과 거짓(False)을 나타내는 자료형
- 자료형의 참과 거짓
 - “Python”: True
 - “”: False
 - []: False
 - 1: True
 - (): False
 - 0: False

변수

변수란?

- 변하는 값
- 파이썬에서 사용하는 변수는 객체를 가리키는 것이라고 할 수 있음
- Ex. >>> a = [1, 2, 3]
 - [1, 2, 3] 값을 가지는 리스트 자료형이 자동으로 메모리에 생성되고, 변수 a는 [1, 2, 3] 리스트가 저장된 메모리의 주소를 가리키게 됨
 - 메모리란, 컴퓨터가 프로그램에서 사용하는 데이터를 기억하는 공간

변수

변수 만드는 방법

- 변수를 만들 때는 =(assignment) 기호를 사용
- 타 프로그래밍 언어(c, java, ...)에서는 변수를 만들 때 자료형을 직접 지정
- 파이썬은 변수에 저장된 값을 스스로 판단하여 자료형을 지정
- 변수명 = 값

제어문

if 조건문이란?, 연산자의 종류

- 조건문: 참과 거짓을 판단하는 문장
- 비교연산자
 - <, >, ==, !=, >=, <=
- and, or, not
 - or: 둘 중 하나만 참이어도 참
 - and: 둘 다 참이어야 참
 - not: 거짓이면 참
- x in s, x not in s
 - in (리스트, 튜플, 문자열)
 - not in(리스트, 튜플, 문자열)
- 조건부 표현식
 - 조건문이 참인 경우 if 조건문 else 조건문이 거짓인 경우

제어문

if - 기본 구조

if 조건문:

수행 문장1

...

else if 조건문:

수행 문장1

...

else:

수행 문장1

...

- 들여쓰기

- if문에 속하는 모든 문장에는 들여쓰기(indentation)를 해야함
- 한 조건문 안에 수행하는 문장은 언제나 같은 너비로 들여쓰기 해야함
- 공백(spacebar), 탭(tab) 어떠한 방식이여도 무관하나, 통일해야 함
- 최근 파이썬 커뮤니티에서는 들여쓰기 시 공백(spacebar) 4개를 사용하는 것을 권장함

제어문

while - 기본 구조

반복문

while 조건문:

수행 문장1

수행 문장2

수행 문장3

...

제어문

for - 기본 구조

반복문

for 변수 in 리스트(또는 튜플, 문자열):

수행 문장1

수행 문장2

...

제어문

for - range()

- range() 함수
 - range(시작 idx, 끝 idx+1)

```
>>> add = 0
```

```
>>> for i in range(1, 11):
```

```
    add = add + i
```

```
>>> print( add )
```

출력결과: ?

함수

함수란?

- 입력값을 가지고 어떤 일을 수행한 다음에 그 결과물을 내어놓는 것
- 개념 정리
 - 함수 정의: 어떤 작업을 수행하는 코드를 모아 이름을 붙이는 것
 - 함수 호출: 함수의 이름을 불러 함수의 내용을 실행하는 것

함수

함수의 필요성

- 반복적인 프로그래밍을 피할 수 있음 (코드 중복 제거)
- 프로그램을 여러 개의 함수로 나누어 작성 시 모듈화로 인해 코드 가독성이 좋아짐
- 유지보수가 쉬워짐 (코드의 재활용성 up)

함수

함수의 구조를 들어가기 전 용어 정리!!

- 예약어: 컴퓨터 프로그래밍 언어에서 이미 문법적인 용도로 사용되고 있기 때문에 식별자로 사용할 수 없는 단어
- 메소드: 클래스 내부에 정의한 함수
- 함수명: 개발자가 임의로 정한 함수의 이름
- 인자/인수: 어떤 함수를 호출시에 전달되는 값
- 매개변수: 전달된 인자를 받아들이는 변수

함수

함수의 구조

```
def 함수명( 매개변수 ):
```

```
    수행 문장1
```

```
    수행 문장2
```

```
    ...
```

문제풀이

코딩 수준 테스트(1차시)

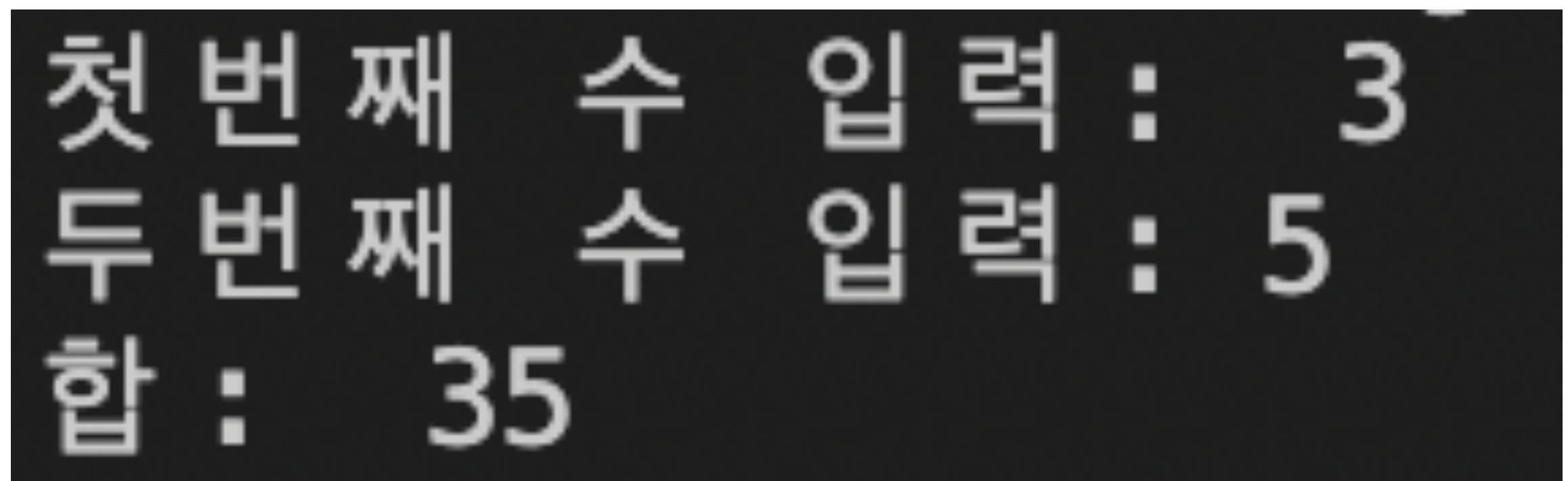
1. 사용자에게 2개의 숫자를 입력받고, 이 2 숫자의 합을 화면에 출력하시오.

- input() 함수를 사용하시오

```
num1 = input("첫번째 수 입력: ")
```

```
num2 = input("두번째 수 입력: ")
```

```
print("합: ", num1 + num2)
```



```
첫 번째 수 입력 : 3
두 번째 수 입력 : 5
합 : 35
```

문제풀이

코딩 수준 테스트(1차시)

2. (1)의 프로그램을 수행하는 '함수'를 만들고, 이 함수를 호출하여 프로그램을 실행하시오.

- 함수 모양: `def sum_two(num1, num2):`

```
첫 번째 수 입력 : 3
두 번째 수 입력 : 5
합 : 35
```

`def sum_two(num1, num2):`

`sum = num1 + num2`

`return sum`

`num1 = input("첫번째 수 입력: ")`

`num2 = input("두번째 수 입력: ")`

`print("합: ", sum_two(num1, num2))`

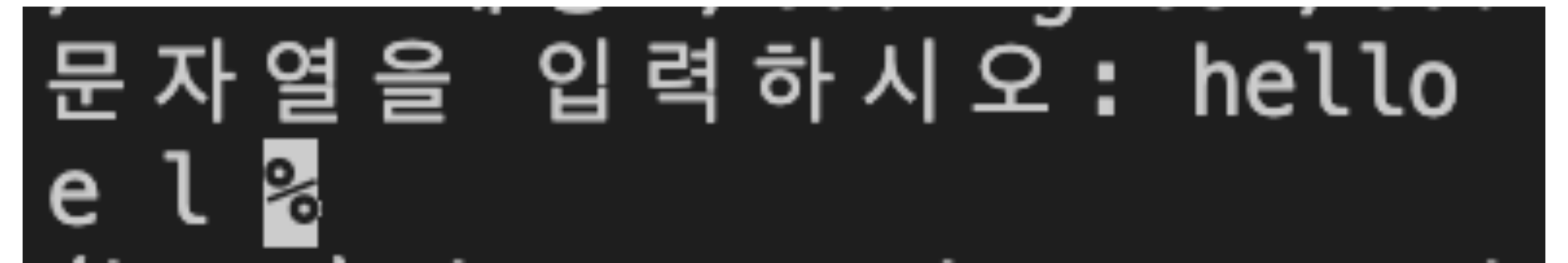
문제풀이

코딩 수준 테스트(1차시)

3. 사용자에게 문자열을 1개 입력 받고, 이 문자열의 짝수 위치에 있는 문자들만을 화면에 출력하시오.

- input() 함수를 사용하시오

- for ... range()문을 사용하시오



```
strInput = input("문자열을 입력하시오: ")
```

```
for i in range(len(strInput)):
```

```
    if i%2 != 0:
```

```
        print(strInput[i], end = ' ')
```


문제풀이

코딩 수준 테스트(1차시)

4. 리스트의 첫번째 항목과 마지막 항목의 값이 같은 지 비교하고, 같으면 True, 다르면 False를 반환하는 함수를 만드시오. 그리고 이 함수를 호출해서 프로그램을 실행하시오.

```
def funcCompare(arrNum):
```

```
    if arrNum[0] == arrNum[-1]:
```

```
        return True
```

```
    else:
```

```
        return False
```

```
arrNum = list(map(int, input().split()))
```

```
print("첫번째 요소와 마지막 요소가 같은지?: ", funcCompare(arrNum));
```

```
1 2 3 4 1
첫번째 요소와 마지막 요소가 같은지?: True
```

```
1 2 3 4 5
첫번째 요소와 마지막 요소가 같은지?: False
```

감사합니다:)