PYTHON TUTORING #2

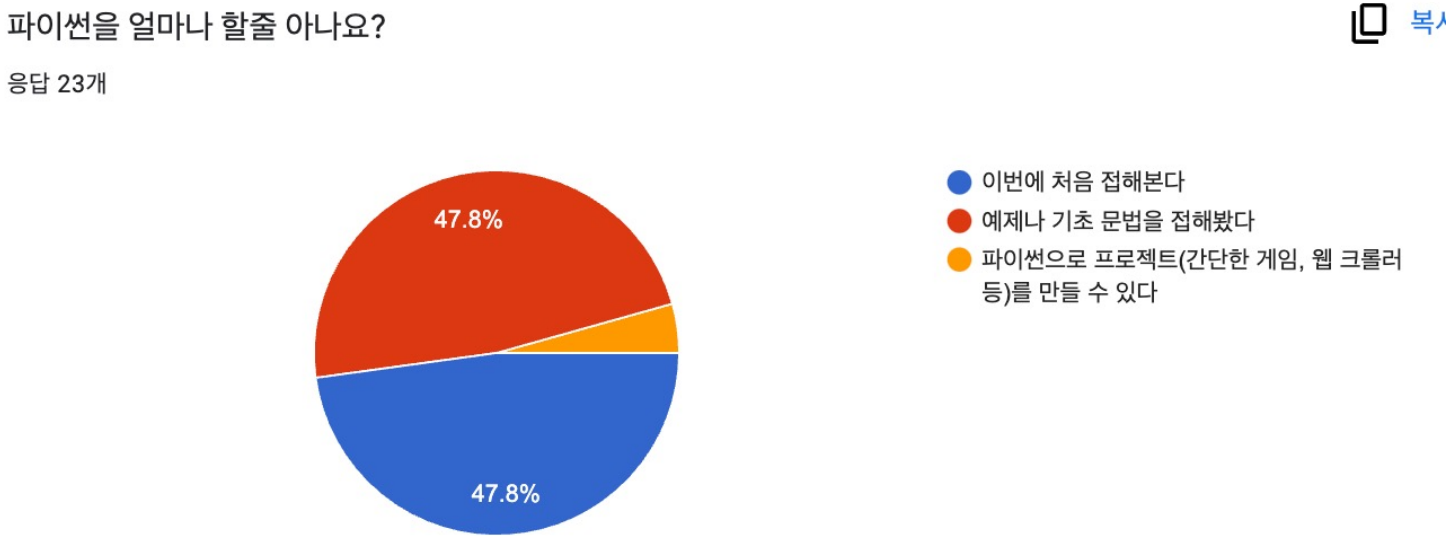
School of Computing, KAIST & 대덕고등학교 빛나리

목차

① 저번 시간 REVIEW

② 리스트, 튜플, 딕셔너리 + 예제

설문조사 결과

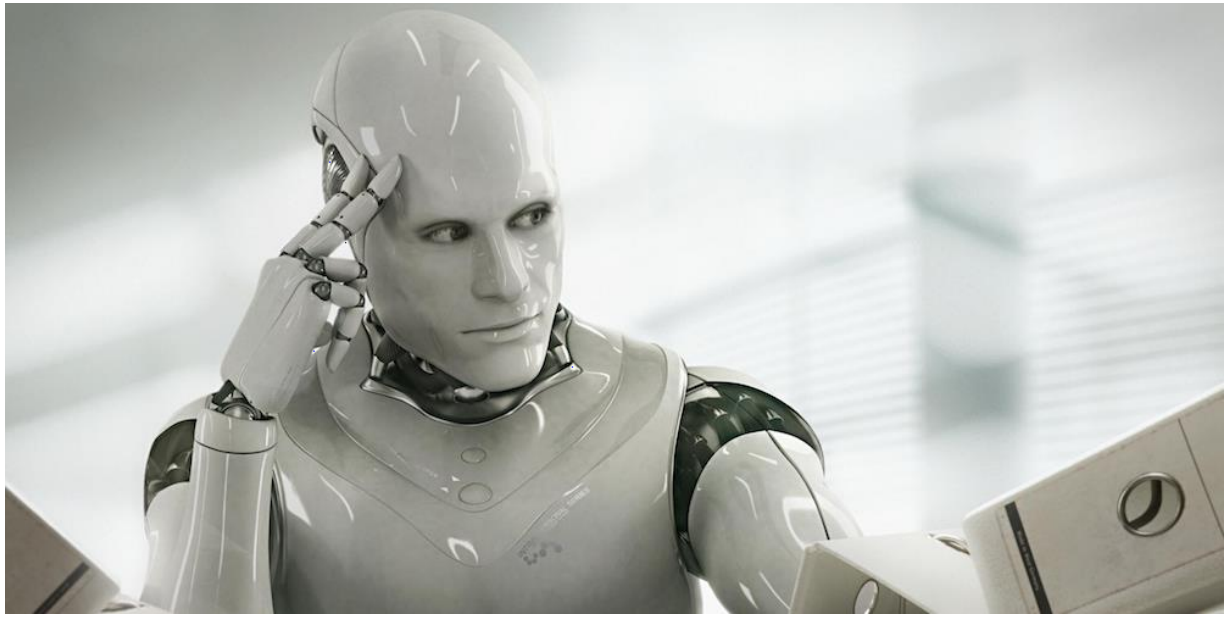


질문 많이 해주세요!
모두에게 큰 도움이 됩니다
피드백: ‘빛나리 프로그래밍 활동 일지’(ex. 이부분 잘 몰라요 등)

프로그래밍이란?

문제들을 해결하기 위한 계획(ex. 파일 찾기)

컴퓨터는 몇 억 번의 단순계산을 1초만에 할 수 있음



BUT 컴퓨터는 단순 작업만 가능

프로그래밍
= 실제 문제 해결 과정을 컴퓨터가 처리할 수 있는 수준으로 나눠서 정리한 것

Why Python?

① 쉽고 간결한 문법

ex) **if** 4 **in** [1,2,3,4]: print("4가 있습니다")

② 개발자를 위한 다양한 도구들

③ ***"인생은 짧다"***

Python에서 사칙연산

왜 하나요? -> 가장 기본적인 연산들

덧셈, 뺄셈, 곱셈, 나눗셈
//는 나누기의 몫
%는 나누기의 나머지
**는 거듭제곱
괄호도 있어요!

예시

```
>>> 1 + 2
>>> 3 – 6

>>> 2 * 7
>>> 7 / 3

>>> 7 // 3
>>> 7 % 3
```

Python 변수

왜 쓰나요? -> 코드의 재활용, 가독성

```
print(100 + 1)
print(100 + 2)
....
print(100 + 10000)
```

Think: 앞에 100을 101로 바꿔야 하는 상황

최대한 변수를 만들지 않고 구현한 이후에,
필요한 것 같으면 그 때 변수 선언하는 방식

예시

```
>>> a = 1 + 2
>>> b = 3 - 6
>>> a
>>> b
>>> a + b
>>> a - b

>>> c = a * b
>>> a
```

Python 변수

변수는 값을 저장하는 공간

= 기호를 통해서 변수에 값을 저장,
추후에 바꿀 수 있음

변수 이름 = 변수에 저장할 값

후에 같은 이름의 변수가 사용되면, 대입된 값으로 명령을 수행하게 된다.

예시

```
>>> a = 1 + 2
>>> b = 3 - 6
>>> a
>>> b
>>> a + b
>>> a - b

>>> c = a * b
>>> print(a)
```

Python 문자열

왜 쓰나요? -> 사람이 읽기 편한 데이터!

사실 컴퓨터 입장에서는 숫자가 더 편해요

“hello” -> 104 101 108 108 111

Spoiler: 곧 배울 List

예시

```
>>> var1 = “Hello”
>>> var2 = “KAIST”

>>> print(var1)
>>> print(var2)

>>> var = var1 + var2
>>> print(var)
```

Python Bool

왜 쓰나요? -> 나중에 배울 if 문

두 수나 문자열 등이 같은지 비교(==)

수가 크거나 같은지 비교(<, <=, >, >=)

여러개도 사용 가능(1 < 2 < 3)

예시

```
>>> var1 = “Hello”
>>> var2 = “KAIST”

>>> var1 == var2
>>> 1 < 3

>>> 1 == 1
>>> 1 <= 2
```

Python Input

왜 쓰나요? -> 숫자야구 프로그램
계산기 웹 브라우저 등등…
사용자 입력을 받기 위해서

함수? -> 특정 목적을 수행하는 코드의 집합
Ex. getBiggestFromList(list1)
스스로 정의 가능

괄호? 읽기 편하게. getBigNumber (a, b, c)

예시

```
>>> a = input("ABC")
ABC
>? 3

>>> b = input("CBA")
CBA
>? "hello"

>>> print(a)
>>> print(b)
```

Int(Input) ?

Input을 Int로 감싸주는 이유는 무엇일까요?

Input 함수를 통해 저장되는 값은 무조건 문자열(str) 이기 때문입니다.

Python의 모든 데이터는 object(객체)로 이뤄져 있어요

- 숫자(int, float)
- “abcd”, “안녕” 같은 문자열(str)

-> 문자열 “13”을 int 자료형으로 바꿔줄게!

-> int(“13”)으로 나타남

예시

```
>>> a = input(“ABC”)
ABC
>? 3

>>> b = input(“CBA”)
CBA
>? “hello”

>>> print(a)
>>> print(b)
```

목차

① 저번 시간 REVIEW

② 리스트, 튜플, 딕셔너리 + 예제

Python 객체!

Python의 모든 데이터는 object(객체)로 이뤄져 있어요

- 숫자(int, float)
- “abcd”, “안녕” 같은 문자열(str)
- True, False를 나타내는 Bool
- [1,2,3,4,5] 같이 여러 값들이 묶여있는 list

등등.....

오늘은 그 다음으로 List, Tuple, Dictionary에 대해 배워보아요

List 자료형

왜 쓰나요?

변수 3개가 있을 때: a,b,c 를 데리고 다니면서 씀
a=100 b = 100 c = 100
변수 100개가 있을 때: 아앗.....
100개를 한번에 들고 있어야겠다! -> List 사용!

예시

```
odd = [1, 3, 5, 7, 9]

empty = []

word = ["even", "odd"]
arr = [1, 2, "number"]

print(odd + word)
print(word * 2)
```

List 자료형

List는 0개 이상의 값을 가지는 자료형

대괄호[]로 감싸져 있으면 List

List 안의 요소는 숫자, 문자열, 리스트
상관없이 모두 가능

List 사이의 연산도 따로 적용

요소의 삽입, 수정이 자유롭다.

예시

```
odd = [1, 3, 5, 7, 9]

empty = []

word = ["even", "odd"]
arr = [1, 2, "number"]

print(odd + word)
print(word * 2)
```

List indexing

List[n]은 n+1번째 요소를 의미한다.

-> 첫 요소는 List[0]에 있음

왜요?
(많은 사람들의 시행착오)
길이가 5일 때, 1 <= index < (5+1) vs 0 <= index < 5

예시

```
arr = [1, 2, "number"]

print(arr[0])
print(arr[1])
print(arr[2])
```

```
arr[0]: 1
arr[1]: 2
arr[2]: "number"
```

예제 – Why list

변수 5개를 만들고 각각 1,2,3,4,5 값을 넣어주세요.
이후 각 변수를 이용해서 2,4,6,8,10 을 출력해주세요.

Hint:

a = 1;b = 2;c = 3;d = 4;e = 5; -> ; 는 컴퓨터가 이해 가능한 줄의 끝.

변수 = 변수 * 2

print(a, b, c, d, e)

(변수가 100개가 된다면?)

예제

[1,3,5,7,9] list를 [2,4,6,8,10] 으로 만들고나서 출력해주세요.

Hint:
l[0] = l[0] + 1 또는 l[0] = 2 -> List
print(l)
for, if 등 쓰시면 안돼요

List slicing

예시: 두개의 리스트를 합쳤을 때, 앞과 뒤로 나누는 경우

List[a:b]는 a+1번째 요소부터
b번째 요소까지를 의미한다.

a, b는 생략 가능, 생략했을 땐 남은 범위 전부 의미

예시

```
arr = [1, 2, "number"]

print(arr[:2])
print(arr[1:])
print(arr[:])
```

arr[:2] = [1, 2]
arr[1:] = [2, “number”]
arr[:] = [1, 2, “number”]

예제

[1,3,5,7,9,”1”,”2”,”3”] list를 [1,3,5,7,9]와 [“1”,”2”,”3”]으로 나눠서 출력해주세요.

Hint:

l[:5] / l[5:] (5는 5번째 요소)

print(l)

List functions

직접 구현하면 귀찮거나 힘든 것.
미리 구현해서 쓰시라고 만들어놨어요

append는 list의 맨 끝에
괄호 안의 요소를 추가한다.

remove는 괄호 안의 요소가 list 안에 있으면
첫 번째로 만나는 요소를 제거한다.

예시

```
arr = [1, 2, "number"]


arr.append(3)
print(arr)


arr.append(2)
print(arr)
arr.remove(2)
print(arr)
```

예제

변수 5개를 입력받아서, 해당 변수 값을 list에 순서대로 넣고 출력해주세요.

Hint:
빈 리스트 -> [] 아니면 list()
Input() -> user 입력받기
l.append(값)

튜플 자료형

List와 비슷하지만 몇 가지가 다르다.

- [] 대신에 () 를 사용함
- 튜플은 요소의 추가, 수정이 안된다.

TypeError: 'tuple' object does not support item assignment

Indexing, slicing 둘 다 가능

왜 써요? -> 더 빨라요. 10억개의 데이터
“일부러 못바꾸게 하고 싶을때” 실수 막기

예시

```
tup   = ( 1 ,  2,  3)

tup2    = (1, )

tup3   = (1, ‘a’, (‘ab’, ‘cd’))
```

딕셔너리 자료형

사전 같아서 딕셔너리!
Why? List와 다르게 key 지정 가능
['lee', '0424'] vs {'name': 'lee', 'birth': '0424'}
실수를 덜 하고 직관적임

Key-Value 쌍 여러 개로 구성되어 있다.
중괄호{}로 둘러싸여 있다.

Key는 변하지 않는 모든 값, value는 모든 값을 사용할 수 있다.

예시

```
dic = {'name' : 'lee',
       'birth' : '0424'
       'age' : 22}


a = {1 : 'abc', 'n' : 33}


  dic['name'] = 'lee'
  dic['age'] = 22
  a[1] = 'abc'
```

딕셔너리 자료형

딕셔너리 쌍 추가하기
a[‘key’] = value

딕셔너리 쌍 제거하기
del a[‘key’] = value

예시

```
>>> a = {1 : ‘abc’, ‘n’ : 33}
>>> a[2] = ‘b’
>>> a
{1: ‘abc’, ‘n’: 33, 2: ‘b’}

>>> del a[‘n’]
>>> a
{1: ‘abc’, 2: ‘b’}
```

딕셔너리 functions

keys는 딕셔너리의 키만,
values는 딕셔너리의 값만 돌려준다.

in은 해당 key가 딕셔너리 안에 있는지 조사한다.

예시

```
>>> a = {1 : 'abc', 'n' : 33}
>>> a.keys()
dict_keys([1, 'n'])

>>> a.values()
dict_values(['abc', 33])

>>> 1 in a
True
```

예제

Input() 함수를 이용하여 ‘name’, ‘age’, ‘birth’를 입력받아 딕셔너리를 만들어보고, 해당 딕셔너리의 name과 birth를 출력해주세요.

Hint:
{} 또는 dict() -> dictionary 선언
d[‘name’] = 값 . ’name’은 key가 됨
Print()

예제

방금 전에 만든 딕셔너리의 value들로 이뤄진 리스트를 만들고,
해당 리스트에서 Birth의 값을 가져와 출력해주세요.

{‘name’: ’lightnari’, ‘age’: 18, ‘birth’: ‘0422’} -> [‘lightnari’, 18, ‘0422’] -> ‘0422’

Hint:

A = dictionary.values()

List(A) (가능해요. dict_values()의 값은 list로 변경 가능한 object)

List1[1] -> 2번째 요소(‘age’)