서울시립대학교 토목공학과

파이썬의 기본 문법부터 딥러닝 맛보기까지

박상수 박사과정

2020년 8월 25일

강의 목차

- **심화 데이터 타입**
  - 리스트, 튜플, 딕셔너리
- **함수와 모듈**
  - Python을 좀더 쉽고 편하게 사용하는 방법
- **Python 연습문제**
  - 지금까지 배운 지식 정리

리스트 (List) #1

리스트: 여러 요소들을 갖는 집합 (컬렉션)

- 요소: 숫자, 문자열
  - 예) 가족 (어머니, 아버지, 본인, 동생)

```
>>> family = ['mother', 'father', 'gentleman', 'sexy lady']
```

  - 리스트의 요소는 숫자, 문자, 빈 리스트 가능

```
a = []     # 빈 리스트
a = ["AB", 10, False]
```

- 인덱싱: 리스트의 특정 한 요소만을 선택하기 위한 방법
  - 첫번째 요소는 리스트 [0], 두번째 요소는 리스트 [1] 처럼 표현

```
a = ["AB", 10, False]
x = a[1]         # a의 두번째 요소 읽기
a[1] = "Test"    # a의 두번째 요소 변경
y = a[-1]        # False
print(a[1])
```

  - 인덱스에 -1, -2 같은 음수 사용 가능 (-1: 현재 리스트의 마지막 요소, -2는 뒤에서 두번째 요소)

리스트 (List) #2

슬라이싱 (Slicing)

- 리스트에서 일부 부분 요소들을 선택하기 위해서 사용
  - 리스트[처음_인덱스:마지막_인덱스], 인덱스 표현에서 부분집합의 범위를 지정

```
a = [1, 3, 5, 7, 10]      # 리스트
x = a[1:3]                # [3,5]
x = a[:2]                 # [1,3]
x = a[3:]                 # [7,10]
```

  - 인덱스는 0부터 시작, 마지막 인덱스는 "마지막 요소의 인덱스 +1"
  - 처음 인덱스가 생략되면 0부터 시작, 마지막 인덱스가 생략되면 끝까지 포함

리스트 요소 추가, 수정 및 삭제

- 리스트 추가: "리스트.append()" 사용

```
a = ["AB", 10, False]
a.append(21.5)     # 추가
a[1] = 11          # 변경
del a[2]           # 삭제
print(a)           # ['AB', 11, 21.5]
```

리스트 (List) #3

리스트 병합과 반복

- 두 개의 리스트를 병합하기 위해서 플러스(+) 사용

```
# 병합
a = [1, 2].
b = [3, 4, 5]
c = a + b
print(c)        # [1, 2, 3, 4, 5]

# 반복
d = a * 3
print(d)        # [1, 2, 1, 2, 1, 2]
```

List Comprehension

- 리스트의 [...] 괄호 안에 For Loop를 사용하여 반복적으로 표현식을 실행해서 리스트 요소를 정의
  - [표현식 for 요소 in 컬렉션 [if 조건식]]
  - 0부터 9까지 숫자들 중 3으로 나눈 나머지가 0인 숫자에 대해 제곱에 대한 리스트 구현

```
list = [n ** 2 for n in range(10) if n % 3 == 0]

print(list)
# 출력: [0, 9, 36, 81]
```

리스트 (List) 실습

빈 리스트를 만들고 1부터 100까지 추가

```
myList=[]

for i in range(100):
  myList._____(i+1)

print(myList)
```

Result

```
[1, 2, 3, 4, ..., 99, 100]
```

중첩리스트 처리하기

- For loop을 사용하여 myList1을 myList2에 복사
  - myList3에 myList1과 myList2의 곱해진 값을 저장 및 출력

```
myList1=[100, 200, 300, 400, 500]
print(myList1)

myList2=[_______________]
print(myList2)

myList3=[___ for __ in myList2]
print(myList3)
```

Result

```
[100, 200, 300, 400, 500]
[100, 200, 300, 400, 500]
[10000, 40000, 90000, 160000, 250000]
```

튜플 (Tuple) #1

리스트와 유사하지만 새로운 요소의 추가, 갱신 및 삭제 불가능

- 튜플은 한번 결정된 요소를 변경할 수 없는 immutable 데이터 타입
  - 튜플을 표현하기 위해서는 둥근 괄호 (...)를 사용, 컴마 (,)로 구분

```
t = (“AB”, 10, False)
print(t)
```

  - 요소가 하나일 경우 요소 뒤에 콤마 (,)를 붙여 명시적으로 튜플 표시

```
t1 = (123)
print(t1)  # int 타입

t2 = (123,)
print(t2)  # tuple 타입
```

- 인덱싱과 슬라이싱

```
t = (1, 5, 10)

# 인덱스
second = t[1]       # 5
last = t[-1]        # 10

# 슬라이스
s = t[1:2]          # (5)
s = t[1:]           # (5, 10)
```

튜플 (Tuple) #2

튜플 병합과 반복

- 리스트와 마찬가지로 병합하기 위해 플러스 (+) 사용, N번 반복하기 위해서는 "튜플*N" 사용

```
# 병합
a = (1, 2)
b = (3, 4, 5)
c = a + b
print(c)   # (1, 2, 3, 4, 5)

# 반복
d = a * 3  # 혹은 "d = 3 * a" 도 동일
print(d)   # (1, 2, 1, 2, 1, 2)
```

튜플 변수 할당

- 튜플 데이터를 변수에 할당할 때, 각 요소를 각각 다른 변수에 할당 가능

```
name = ("John", "Kim")
print(name)
# 출력: ('John', 'Kim')

firstname, lastname = ("John", "Kim")
print(lastname, ",", firstname)
# 출력: Kim, John
```

튜플 (Tuple) 실습

튜플은 정말로 변경이 불가능할까 ?

```
# List
number1 = [1, 2, 3, 4]
number1[0]=100
print(number1)

# Tuple
number2 = (1, 2, 3, 4)
number2[0]=100
print(number2)
```

Result

```
[100, 2, 3, 4]
Traceback (most recent call last):
  File "test.py", line 8, in <module>
    number2[0]=100
TypeError: 'tuple' object does not support item assignment
```

튜플을 리스트로 만들고, 리스트를 튜플로 만들기

- 튜플과 리스트는 변환이 가능
  - `tuple(리스트_객체)`, `list(튜플_객체)`

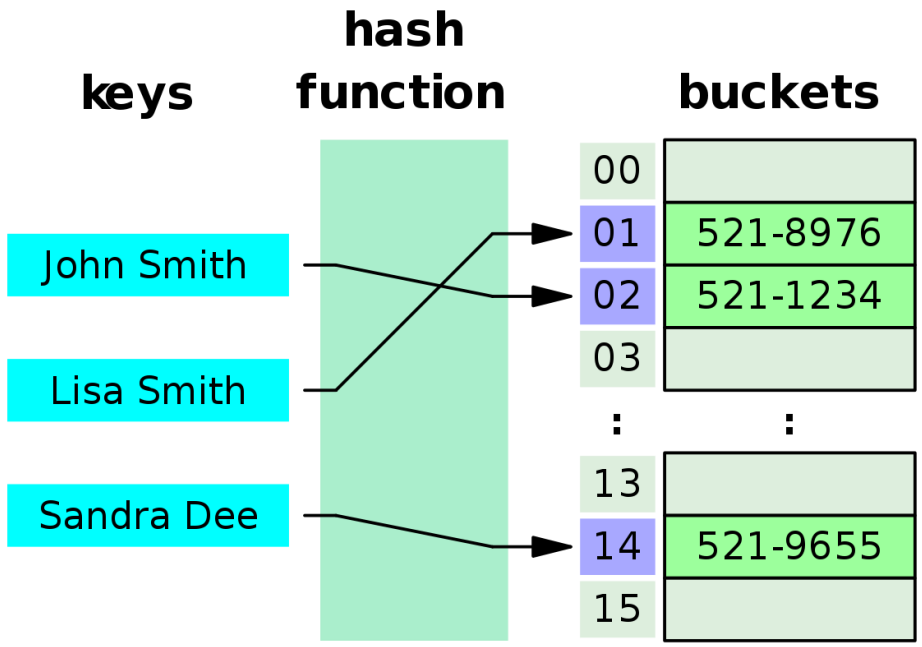
```
# List to Tuple
a = [1, 2, 3]
a=_____(a)
a[0]=100

# Tuple to List
b = (4, 5, 6)
b=____(b)
b[0]=400
```

Result

```
Traceback (most recent call last):
  File "test.py", line 4, in <module>
    a[0]=100
TypeError: 'tuple' object does not support item assignment
```

딕셔너리 (Dictionary) #1

- **딕셔너리: "키 (Key)와 값 (Value) 쌍을 요소로 갖는 컬렉션 "**
  - 키 (Key)로 신속하게 값 (Value)을 찾아내는 해시 테이블 (Hash Table) 구조
  - 딕셔너리의 요소들은 리터럴 "{...}" 를 사용하여 컬렉션 표현
    - 각 요소들은 **"Key:Value"** 쌍으로 구성되어 있으며, 요소 간은 콤마로 구분
    - 딕셔너리를 생성하기 위해 dict() 생성자 (Key-Value 쌍 입력) 또는 dict(key=value, ...) 식으로 생성 가능



키와 값의 예: 키 (사람 이름), 값 (전화번호)

```
scores = {"철수": 90, "민수": 85, "영희": 80}
v = scores["민수"]   # 특정 요소 읽기
scores["민수"] = 88  # 쓰기
print(scores["민수"])
```

리터럴을 이용한 딕셔너리 생성

```
# 1. Tuple List로부터 dict 생성
persons = [('김기수', 30), ('홍대길', 35), ('강찬수', 25)]
mydict = dict(persons)

age = mydict["홍대길"]
print(age)   # 35

# 2. Key=Value 파라미터로부터 dict 생성
scores = dict(a=80, b=90, c=85)
print(scores['b'])  #90
```

dict() 생성자와 Key=Value 파라미터를 이용한 방법

딕셔너리 (Dictionary) #2

딕셔너리 수정 (추가, 삭제, 읽기)

- 딕셔너리 요소를 수정하기 위해서는 **"Dictionary[키]=새로운 값"** 형식으로 입력
- 삭제하기 위해서는 "del 요소 " 를 사용하여 특정 요소를 삭제

```
scores = {"철수": 90, "민수": 85, "영희": 80}
scores["민수"] = 88   # 수정
scores["길동"] = 95   # 추가
del scores["영희"]
print(scores)
# 출력 {'철수': 90, '길동': 95, '민수': 88}
```

- 딕셔너리에 있는 값들을 출력하기 위해서는 다음과 같이 For Loop을 사용

```
scores = {"철수": 90, "민수": 85, "영희": 80}

for key in scores:
    val = scores[key]
    print("%s : %d" % (key, val))
```

Result

```
철수 : 90
민수 : 85
영희 : 80
```

딕셔너리 (Dictionary) 실습문제

- **다음 소스 코드를 완성하여 게임의 체력/이동 속도가 출력되도록 만드세요**
  - 각 키에 따른 값을 출력하기 위해서는 **Camille [키값]** 형식으로 사용

```
camille = {
    'health': 575.6,
    'health_regen': 1.7,
    'mana': 338.8,
    'mana_regen': 1.63,
    'melee': 125,
    'attack_damage': 60,
    'attack_speed': 0.625,
    'armor': 26,
    'magic_resistance': 32.1,
    'movement_speed': 340
}

print(                    )
print(                    )
```

Result

```
575.6
340
```

강의 목차

- 심화 데이터 타입
  - 리스트, 튜플, 딕셔너리
- **함수와 모듈**
  - Python을 좀더 쉽고 편하게 사용하는 방법
- Python 연습문제
  - 지금까지 배운 지식 정리

함수 (Function) #1

일정한 작업을 수행하는 코드 블록

- 보통 반복적으로 계속 사용되는 코드들을 함수로 정의하여 사용
- Python에서 함수는 def 키워드를 사용하여 정의

```
def 함수명(입력 파라미터):
    문장1
    문장2
    [return 리턴값]
```

```
def sum(a, b):
    s = a + b
    return s

total = sum(4, 7) # 호출자
print(total)
```

- **Default/Named parameter**
  - **Default**: 함수에 전달되는 입력 파라미터 중 호출자아가 전달하지 않으면 디폴트로 지정된 값 사용
  - 함수를 호출할 때, 보통 함수에 정의된 순서에 따라 입력 파라미터를 전달
    - **Named**: "파라미터명=파라미터값"과 같은 형식으로 파라미터 전달 가능

Default

```
def calc(i, j, factor = 1):
    return i * j * factor

result = calc(10, 20)
print(result)
```

Named

```
def report(name, age, score):
    print(name, score)

report(age=10, name="Kim", score=80)
```

함수 (Function) #2

가변길이 파라미터

- 함수의 입력 파라미터 개수를 미리 알 수 없거나, 파라미터의 개수가 변하는 경우 사용
- 파라미터 옆에 *를 붙여 가변길이임을 표시

```
def total(*numbers):
    tot = 0
    for n in numbers:
        tot += n
    return tot

t = total(1,2)
print(t)
t = total(1,5,2,6)
print(t)
```

연습문제 (구구단)

- 반복문 (while)과 함수 (multi)를 사용한 구구단 프로그램

```
def multi(a):
   i = 1
   (________ 9):
     print(a, ' * ', i, ' = ', a * i)
     i = i + 1

multi(9)
```

Result

```
9 * 1 = 9
9 * 2 = 18
.
.
.
9 * 9 = 81
```

모듈 (Module) #1

모듈: 복잡한 프로그램을 작성하기 위한 방법

- 전체적인 모습에서부터 작은 기능까지 구상하고, 만들고, 오류를 수정하는 것은 쉽지 않음
- Python에서는 이러한 문제를 해결하기 위해 모듈이라는 개념을 사용
  - **모듈 (Module): 프로그램의 꾸러미**

pi value

```
>>> import math                    # math 모듈을 불러온다
>>> math.pi                        # math 모듈의 변수 pi의 값은?
3.1415926535897931
```

calendar

```
Python 3.6.10 (default, Dec 19 2019, 23:04:32)
[GCC 5.4.0 20160609] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> import calendar
>>> calendar.prmonth(2013, 7)
     July 2013
Mo Tu We Th Fr Sa Su
 1  2  3  4  5  6  7
 8  9 10 11 12 13 14
15 16 17 18 19 20 21
22 23 24 25 26 27 28
29 30 31
```

모듈 (Module) #2

모듈 가져오기 (import)

- Import를 사용하여 모듈을 불러올 수 있음
  - (1) 모듈 전체를 불러오거나 (2) 모듈 내에서 필요한 것만 가져오는 방법

```
import 모듈

from 모듈 import 변수나 함수
```

- Python의 대표적인 모듈 (Random)
  - Random: 난수 발생을 위한 모듈
  - `random()` 함수는 0이상 1미만의 숫자 중에서 하나의 값을 출력 (실수)
  - 특정 범위의 정수 중 하나를 무작위로 얻기 위해서는 `randrange()` 함수 사용

```
>>> import random
>>> random.random()
>>> 0.90389642027948769

>>> random.randrange(1,7)
>>> 6
```

강의 목차

- 심화 데이터 타입
  - 리스트, 튜플, 딕셔너리
- 함수와 모듈
  - Python을 좀더 쉽고 편하게 사용하는 방법
- **Python 연습문제**
  - 지금까지 배운 지식 정리

Python 연습문제

소수 (Prime number)를 판별하는 프로그램

- 소수: 1과 자기 자신 외에 양의 약수가 없는 1보다 큰 자연수
  - 예) 자연수 5를 자신보다 작은 수인 [2, 3, 4]로 나눠 보면 [1, 2, 1] 나머지 발생
  - **소수를 판별하는 방법은 특정 숫자 보다 작은 수로 나눠서 나머지 발생 여부를 확인**

```
def prime_number(number):           # number를 입력 받아, 소수인지 아닌지 파악하는 함수
    if ________ != 1:               # number가 1이 아니면
        for f in _______________:   # 2, 3, ..., (number-1) 까지의 인수에 대해서
            if number%f == 0:       # number가 위의 인수 중에 하나로 나눠지면 (나머지가 0이면)
                return False        # 소수가 아님
    else:                           # number가 1이라면
        return False                # 소수가 아님
    # number가 1이 아니면서, 2부터 (number-1) 까지의 수로 나눠지지 않으므로
    # 소수로 판별
    return True

num = 7
if prime_number(int(num)):
    print("TRUE")
else:
    print("FALSE")
```

감사합니다