

Big Data Analytics Programming

Week-11. Review

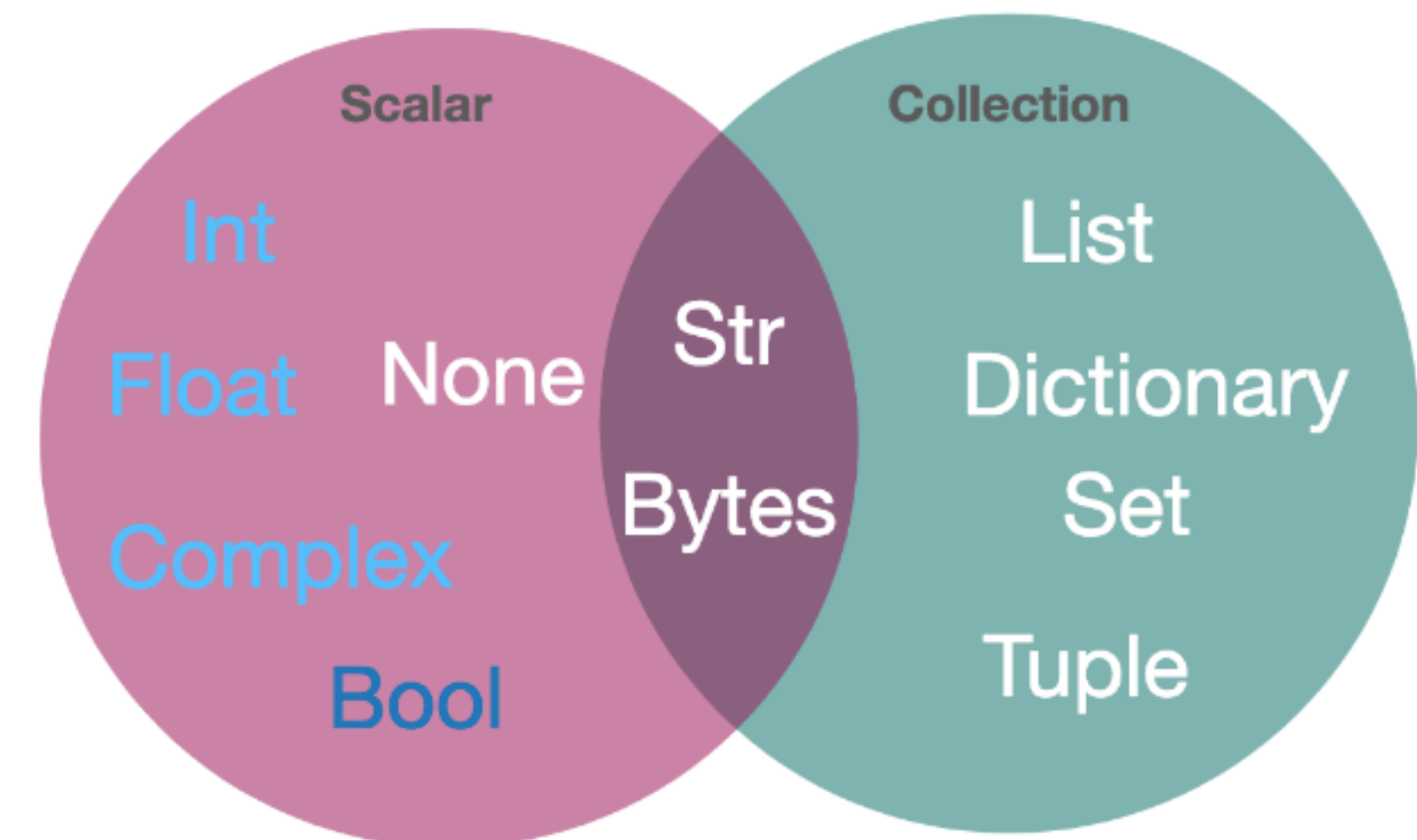
Jungwon Seo, 2021-Spring

1. 변수

파이썬에서의 변수

변수란?

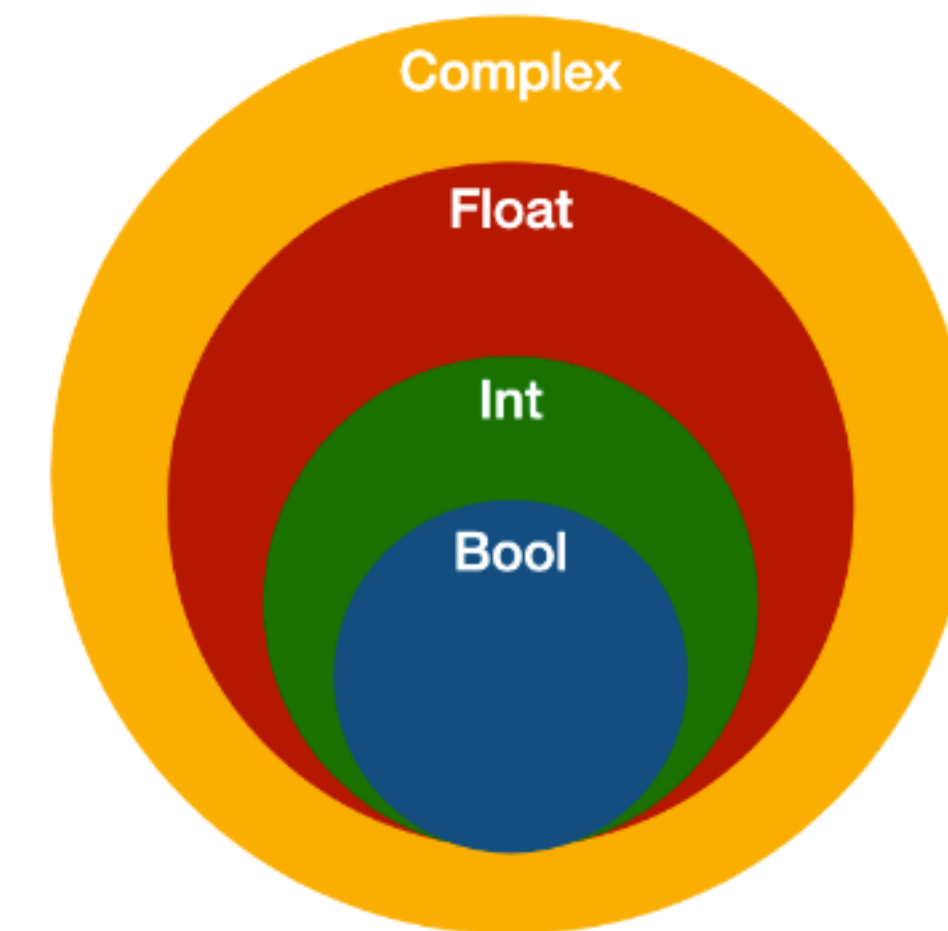
- 변수 (Variable)
 - 메모리에 값을 저장하기 위한 명명된 위치
- Scalar 형태와 Collection 형태로 나눌 수 있음
 - 단일 값을 갖는 Scalar 형태
 - 복수의 값들을 갖을 수 있는 Collection 형태
 - Collection 형태는 자료구조(Data Structure)로도 불림



파이썬에서의 변수

숫자형 변수

- 숫자형 변수의 연산들
 - +, -, *, /: 사칙연산
 - **: 지수연산
 - % : 나머지 구하기
 - //: 몫 구하기
- 숫자형 변수간의 형변환
 - int(a), float(a), bool(a) 와 같은 식으로 casting 함수를 적용
 - 형변환 없이 연산 적용시, 상위 형으로 자동 변형
 - 나누기의 경우 int, int 연산이어도 float으로 자동변형



2. 자료구조

자료구조

문자열 String

- 문자열
 - 쌍따옴표(") 또는 따옴표(')로 감싸진 값
- 문자열의 특징
 - 대소문자가 다르게 인식된다: 'A' != 'a'
 - 숫자 1과 문자열 1은 다르다: 1 != '1'
 - + 연산자는 두 string을 연결하는 역할을 한다: 'he' + 'llo' = 'hello'
 - 문자열간의 *, -, / 연산자는 지원되지 않는다.
 - 문자열 * 숫자는 해당 문자를 숫자만큼 생성한다: 'a'*5 = 'aaaaa'
 - 인덱스로 각각의 문자를 접근할 수 있다: 'hello'[3] => 'l'

0	1	2	3	4
H	E	L	L	O

자료구조

또는 Collection 타입

- 복수의 값들을 저장하기 위한 자료구조들
- List
 - [] 로 감싸지고, 각각의 값은 쉼표로 구분
 - 인덱스로 접근
- Dictionary
 - {}로 감싸지고, key-value 형태가 ':' 기준으로 표현
 - Key로 접근
- Set
 - {}로 감싸지고, 각각의 값은 쉼표로 구분
 - 중복된 값을 허용하지 않음, for 문과 같은 반복자로 접근 해야함
- Tuple
 - ()로 감싸지고, 각각의 값은 쉼표로 구분
 - 인덱스로 접근, 수정 불가

자료구조

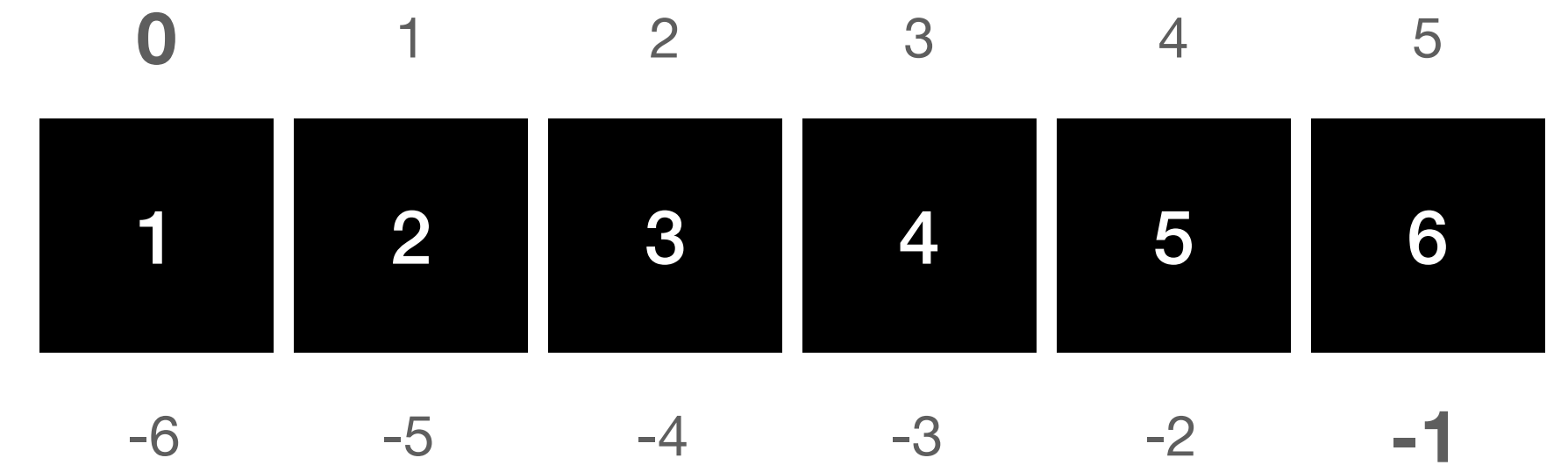
List

- Create
 - `my_list = []` 또는 `my_list = list()`, 만약 초기값(initial value)를 제공한다면, `my_list = [1,2,3]`
- Read
 - list 전체에 대한 접근은 list 명 그대로: `print(my_list)`, `temp_list = my_list`
 - 원소에 대한 접근은: `my_list[index]`, 이때 index는 해당 원소의 위치 (0부터 시작, 정수)
 - 예) `a = [1, 2, 3, 4]` 일 때 `a[3] => ?`
- Update
 - list 전체를 변경하는 경우는 변수 재할당 : `my_list = ['a','b','c']`
 - 원소에 대한 변경은: `my_list[index] = 'k'`
 - 원소 추가는: `my_list.append('x')`
 - Concatenation은 : `[1,2,3] + ['a','b','c'] => [1, 2, 3, 'a', 'b', 'c']`
- Delete
 - list 전체에 대한 delete는: `del my_list`
 - n번째 원소에 대한 delete는 : `del my_list[n]`

자료구조

List

- Python의 List의 경우에는 Subset에 접근 가능
 - 예) `a = [1,2,3,4,5,6]` 일때 앞에 3개만 추출 하려는 경우?
- `list[start:end]`
 - start는 시작하려는 index, end는 마지막의 다음 index
 - `a = [1,2,3,4,5,6]` 일때, `a[0:3]`은 0,1,2 index에 대한 접근 => `[1,2,3]`
 - 만약 start 부분 또는 end 부분을 생략한다면, `list[:] == list [0 : len(list)]`
- `list[-start:-end]`
 - 음수의 index를 넣는 경우, "뒤에서부터 n번째"의 의미
 - 예) `a[-1]`는 맨 뒤의 원소, `a[-3]`은 뒤에서 세번째 원소
 - 예) `a[1:-3]`은? => `[2,3]`



자료구조

Dictionary

- Create
 - `my_dict = { }` 또는 `my_dict = dict()`, 만약 초기값(initial key-value)를 제공한다면, `my_dict = { 'a':1 }`
- Read
 - dictionary 전체에 대한 접근은 dictionary 명 그대로: `print(my_dict), temp_dict = my_dict`
 - Value에 대한 접근은: `my_dict[key]`, 이때 key는 dictionary를 정의 할 때 사용한 Key
- Update
 - dictionary 전체를 변경하는 경우는 변수 재할당 : `my_dict = { "a": 3 }`
 - Value에 대한 변경은: `my_dict[key] = value`
 - Key-Value 추가: `my_dict[new_key] = value`
- Delete
 - dictionary 전체에 대한 delete는: `del my_dict`
 - Key에 대한 delete는 : `del my_list[key]`
 - 이때, Mapping 된 Value도 같이 소멸

자료구조

Dictionary

- Dictionary의 전체 데이터에 대한 접근은 크게 세가지
- 전체 Key 출력
 - `my_dict.keys()`
 - 출력: `dict_keys(['name', 'age', 'height', 'retired'])` => list처럼
- 전체 Value 출력
 - `my_dict.values()`
 - 출력: `dict_values(['손흥민', 29, 183, False])` => list 처럼
- 전체 Item 출력
 - `my_dict.items()`
 - 출력: `dict_items([('name', '손흥민'), ('age', 29), ('height', 183), ('retired', False)])`

4. 조건문과 반복문

조건문과 반복문

조건문

- **if, elif, else**의 키워드로 조건이 만족하는 상황에서만 하위 코드를 실행
- 조건은 True 또는 False 형태로 나오는 값 또는 연산
 - Else는 뒤에 조건이 없음

Team True!! 🧛

Team False!! 🧛

True	False
All numbers except 0	0 0.0
All collections except collection with no element	None []
All functions	() {}

```
score = -5
if score >= 100:
    score = 100
elif score <= 0 :
    score = 0
```

```
score = 70
grade = None
if score >= 50:
    grade = "Pass"
else:
    grade = "Fail"
```

조건문과 반복문

반복문

- 반복문 while

- 특정 조건을 만족하는 한 하위 코드를 반복해서 실행

```
budget = 100
while budget > 0 :
    budget -= buy()
print("No more money")
```

- 반복문 for

- Collection 타입의 데이터의 내부 원소 개수 만큼 반복분을 실행

- 반복문 제어

- Break: 반복문 종료
- Continue: 현재 지점까지만 실행하고 다음 턴으로 넘김

```
my_list = ['a','b','c']
for ele in my_list:
    print(ele)
```

```
for i in range(0,10):
    print(i)
```

5. 함수와 클래스

함수와 클래스

함수 function

- 코드의 가독 성 및 효율 성을 높이기 위해, 특정한 연산들을 한 곳에 모아둔 문법

- 구성요소

- 함수 이름
- 함수 파라미터 (optional)
- 함수 내부 연산
- 함수 반환 값 (optional)

- 주의 할 점

- 함수 내부에서의 변수와 함수 외부에서의 변수는 동작하는 영역이 다름
- 함수 내부에서, 함수 외부 값을 변형하기 위해서는 : global 키워드 사용

```
def outer_func():  
    x = 10  
    def inner_func():  
        x = 3  
        print(x)  
    inner_func()  
    print(x)
```

```
x = 5  
outer_func()  
print(x)
```


함수와 클래스

클래스 Class

- 변수가 상태, 함수가 행동을 표현한다면, 클래스는 둘 다

- class의 구성요소

- method/function: 클래스 내에 있는 함수
- attribute/variable: 클래스 내에 있는 변수
- 생성자
 - 최초 클래스를 인스턴스화 할때 실행되는 부분
 - `__init__`
- 부모클래스
 - 클래스 이름 오른쪽에 괄호와 함께 제공이되는 클래스

```
class Mammal:  
    def __init__(self):  
        pass
```

```
class Human(Mammal):  
    def __init__(self,date_of_birth, name, nationality):  
        self.date_of_birth=date_of_birth  
        self.name = name  
        self.nationality = nationality  
    def think(self):  
        pass  
    def move(self):  
        pass
```

```
jungwon = Human("900302", "서중원", "대한민국")
```

6. 데이터

데이터

데이터 수집

- 파일 읽기관련 함수 및 라이브러리
 - 일반 텍스트 파일: open
 - JSON 파일: json 라이브러리
 - CSV: Pandas
 - HTML: BeautifulSoup
- 네트워크 통신을 통한 파일 읽기
 - requests 라이브러리
 - 데이터베이스의 경우 해당 데이터 베이스와 관련된 라이브러리: 예) mysql connector

데이터 크롤링

- 웹사이트에 있는 정보를 프로그래밍 적으로 수집하는 과정
- Python을 이용한 크롤링
 - 여러 방법이 있지만, requests를 이용한 해당 웹사이트 호출 후 내용 파싱(parsing)
 - 정적 웹사이트: 주소(URL) 그대로 입력을 해서 가져오면 웹사이트에서 보이는 콘텐츠가 그대로 가져와지는 경우
 - 동적 웹사이트: 주소(URL)를 기반으로 requests를 통해 가져온 결과와 웹사이트에서 보여지는 결과가 다른경우
 - 추가적인 URL을 찾아야 함: 크롬 개발자도구
 - header값을 조작하여, 인증 및 기타 보안 관련 장애물들을 우회

7. 데이터 전처리

데이터 전처리

데이터 종류별 전처리 방법

- 텍스트 데이터 전처리
 - 원형복원, 고빈출/저빈출어 제거, 불용어 제거
- 이미지 데이터 전처리
 - 리사이즈: 가로, 세로, 채널, Cropping
- 테이블 형태 데이터 전처리
 - 결측값 제거
 - Noise, Outlier 제거
 - 숫자형 데이터: 정규화 (min-max 스케일링)
 - 범주형 데이터: 인코딩 (one-hot 인코딩)

8. EDA

탐색적 데이터 분석

데이터 파악을 위한 분석

- 시각적 수치적 요약 데이터로 전체 데이터를 다양한 각도로 분석
- 요약 통계
 - 수치형 데이터: 평균, 최대, 최소, 중위값, 분산, 표준편차
 - 명목형 데이터: 최빈값, 빈도수
- 시각화
 - 다양한 차트를 이용

EDA

상관관계 분석

- 상관관계 분석

- 특성간의 상관관계를 비교하여 특성들간의 관계를 수치화한다
 - 상관관계의 범위 -1.0~1.0
 - 양의 상관관계: 한 값이 증가하면 다른 값도 증가
 - 음의 상관관계: 한 값이 증가하면 다른 값은 감소
- 단, 높은 상관관계를 보인다고 해서 반드시 추후 모델 학습에서 더 좋은 성능을 보장하는 것은 아님

E.O.D