문자열과
텍스트 파일 데이터 다루기

배효철
hcbae@pospot.kr

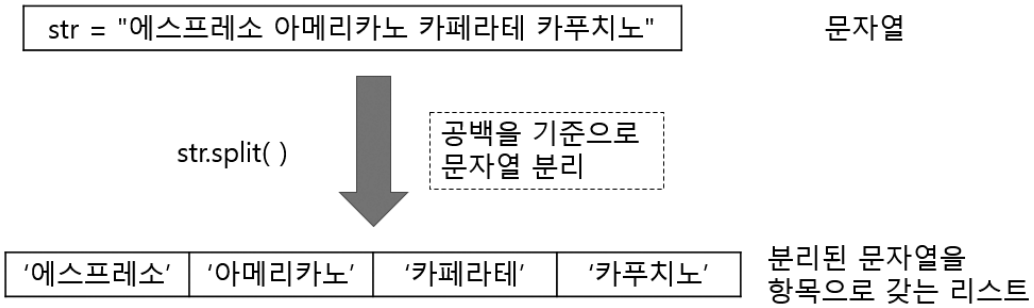
문자열 다루기

- 큰따옴표(")나 작은따옴표(') 안에 들어 있는 문자의 집합
- 텍스트 파일의 내용은 문자열이 됨
- 문자열 처리
  - 문자열 분리
  - 불필요한 문자열 제거
  - 문자열 연결 등

문자열 다루기

- 문자열 분리하기
  - split() 메서드 이용

```
str.split([sep])
```



  - split() 사용 예

```
In: coffee_menu_str = "에스프레소,아메리카노,카페라테,카푸치노"
    coffee_menu_str.split(',')
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

```
In: "에스프레소,아메리카노,카페라테,카푸치노".split(',')
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

문자열 다루기

- split() 사용 예

```
In: "에스프레소 아메리카노 카페라테 카푸치노".split(' ')
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

```
In: "에스프레소 아메리카노 카페라테 카푸치노".split()
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

```
In: "  에스프레소 ₩n₩n 아메리카노 ₩n 카페라테  카푸치노 ₩n₩n".split()
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

```
In: "에스프레소 아메리카노 카페라테 카푸치노".split(maxsplit=2)
Out: ['에스프레소', '아메리카노', '카페라테 카푸치노']
```

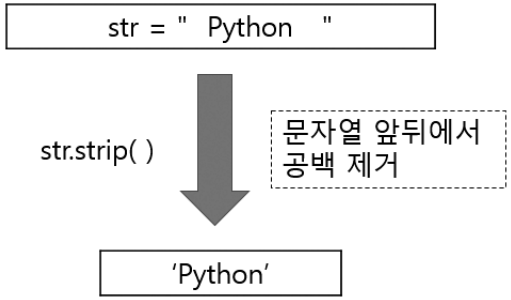
```
In: phone_number = "+82-01-2345-6789"   # 국가 번호가 포함된 전화번호
    split_num = phone_number.split("-", 1)   # 국가 번호와 나머지 번호 분리
    print(split_num)
    print("국내전화번호: {0}".format(split_num[1]))

Out: ['+82', '01-2345-6789']
     국내전화번호: 01-2345-6789
```

문자열 다루기

- 필요 없는 문자열 삭제하기
  - strip() 메서드 이용

```
str.strip([chars])
```



  - strip() 사용 예

```
In: "aaaaPythonaaa".strip('a')
Out: 'Python'
```

```
In: test_str = "aaabbPythonbbbaa"
    temp1 = test_str.strip('a') # 문자열 앞뒤의 'a' 제거
    temp1
Out: 'bbPythonbbb'
```

문자열 다루기

- strip() 사용 예

```
In: temp1.strip('b') # 문자열 앞뒤의 'b' 제거
Out: 'Python'
```

```
In: test_str.strip('ab') # 문자열 앞뒤의 'a'와 'b' 제거
Out: 'Python'
```

```
In: test_str.strip('ba')
Out: 'Python'
```

```
In: test_str_multi = "##***!!!##....  Python is powerful.!...  %%!#..   "
    test_str_multi.strip('*.#! %')
Out: 'Python is powerful'
```

```
In: "  Python   ".strip(' ')
Out: 'Python'
```

```
In: "₩n  Python  ₩n₩n".strip(' ₩n')
Out: 'Python'
```

```
In: "₩n  Python  ₩n₩n".strip()
Out: 'Python'
```

문자열 다루기

- strip() 사용 예

```
In: "aaaBallaaa".strip('a')
Out: 'Ball'
```

```
In: "₩n This is very ₩n fast. ₩n₩n".strip()
Out: 'This is very ₩n fast.'
```

```
In: str_lr = "000Python is easy to learn.000"
    print(str_lr.strip('0'))
    print(str_lr.lstrip('0'))
    print(str_lr.rstrip('0'))
Out: Python is easy to learn.
     Python is easy to learn.000
     000Python is easy to learn.
```

```
In: coffee_menu = "  에스프레소, 아메리카노,    카페라테   , 카푸치노  "
    coffee_menu_list = coffee_menu.split(',')
    coffee_menu_list
Out: ['  에스프레소', ' 아메리카노', '    카페라테   ', ' 카푸치노  ']
```

문자열 다루기

- strip() 사용 예

```
In: coffee_list = [] # 빈 리스트 생성
    for coffee in coffee_menu_list:
        temp = coffee.strip() # 문자열의 공백 제거
        coffee_list.append(temp) # 리스트 변수에 공백이 제거된 문자열 추가

    print(coffee_list) #최종 문자열 리스트 출력
Out: ['에스프레소', '아메리카노', '카페라테', '카푸치노']
```

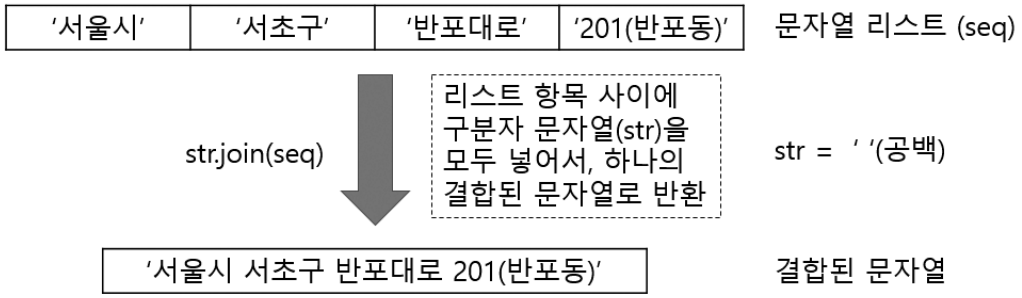
문자열 다루기

- 문자열 연결하기
  - 더하기 연산자(+)로 연결

```
In: name1 = "철수"
    name2 = "영미"
    hello = "님, 주소와 전화 번호를 입력해 주세요."
    print(name1 + hello)
    print(name2 + hello)
Out: 철수님, 주소와 전화 번호를 입력해 주세요.
     영미님, 주소와 전화 번호를 입력해 주세요.
```

  - join() 메서드 이용

```
str.join(seq)
```



문자열 다루기

- join() 메서드 이용 예

```
In: address_list = ["서울시", "서초구", "반포대로", "201(반포동)"]
    address_list
Out: ['서울시', '서초구', '반포대로', '201(반포동)']
```

```
In: a = " "
    a.join(address_list)
Out: '서울시 서초구 반포대로 201(반포동)'
```

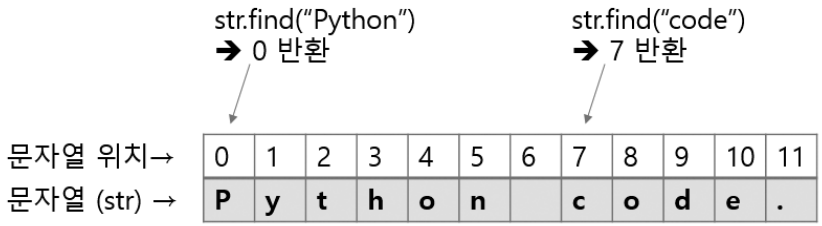
```
In: " ".join(address_list)
Out: '서울시 서초구 반포대로 201(반포동)'
```

```
In: "*^-^*".join(address_list)
Out: '서울시*^-^*서초구*^-^*반포대로*^-^*201(반포동)'
```

문자열 다루기

- 문자열 찾기
  - find() 메서드 이용

```
str.find(search_str)
```




```
In: str_f = "Python code."
    print("찾는 문자열의 위치:", str_f.find("Python"))
    print("찾는 문자열의 위치:", str_f.find("code"))
    print("찾는 문자열의 위치:", str_f.find("n"))
    print("찾는 문자열의 위치:", str_f.find("easy"))
Out: 찾는 문자열의 위치: 0
     찾는 문자열의 위치: 7
     찾는 문자열의 위치: 5
     찾는 문자열의 위치: -1
```

문자열 다루기

- find() 메서드에 시작 위치(start)와 끝 위치(end)를 지정해 검색 범위를 설정

```
str.find(search_str, start, end)
```

- find() 메서드에 시작 위치(start)만 지정해 검색 범위를 설정

```
str.find(search_str, start)
```

```
In: str_f_se = "Python is powerful. Python is easy to learn."
    print(str_f_se.find("Python", 10, 30)) # 시작 위치(start)와 끝 위치(end) 지정
    print(str_f_se.find("Python", 35)) # 찾기 위한 시작 위치(start) 지정
Out: 20
     -1
```

- count()로 문자열 일치 횟수 반환

```
str.count(search_str)
str.count(search_str, start)
str.count(search_str, start, end)
```

문자열 다루기

- count()로 문자열 일치 횟수 반환

```
In: str_c = "Python is powerful. Python is easy to learn. Python is open."
    print("Python의 개수는?:", str_c.count("Python"))
    print("powerful의 개수는?:", str_c.count("powerful"))
    print("IPython의 개수는?:", str_c.count("IPython"))
Out: Python의 개수는?: 3
     powerful의 개수는?: 1
     IPython의 개수는?: 0
```

문자열 다루기

- 문자열이 특정 문자열로 시작하는지 끝나는지 검사
  - startwith(), endwith() 메서드 이용

```
str.startswith(prefix)
str.startswith(prefix, start)
str.startswith(prefix, start, end)
```

```
str.endswith(suffix)
str.endswith(suffix, start)
str.endswith(suffix, start, end)
```

  - startwith(), endwith() 메서드 이용 예

```
In: str_se = "Python is powerful. Python is easy to learn."
    print("Python으로 시작?:", str_se.startswith("Python"))
    print("is로 시작?:", str_se.startswith("is"))
    print(".로 끝?:", str_se.endswith("."))
    print("learn으로 끝?:", str_se.endswith("learn"))
Out: Python으로 시작?: True
     is로 시작?: False
     .로 끝?: True
     learn으로 끝?: False
```

문자열 다루기

- 문자열 바꾸기
  - replace() 메서드 이용

```
str.replace(old, new[, count])
```

  - replace() 메서드 이용 예

```
In: str_a = 'Python is fast. Python is friendly. Python is open.'
    print(str_a.replace('Python', 'IPython'))
    print(str_a.replace('Python', 'IPython', 2))
Out: IPython is fast. IPython is friendly. IPython is open.
     IPython is fast. IPython is friendly. Python is open.
```

```
In: str_b = '[Python] [is] [fast]'
    str_b1 = str_b.replace('[', '') # 문자열에서 '['를 제거
    str_b2 = str_b1.replace(']', '') # 결과 문자열에서 다시 ']'를 제거
    print(str_b)
    print(str_b1)
    print(str_b2)
Out: [Python] [is] [fast]
     Python] is] fast]
     Python is fast
```

문자열 다루기

- 문자열의 구성 확인
  - 문자열이 숫자 또는 문자로만, 숫자와 문자가 모두 포함돼 있는지, 로마자 알파벳 대문자로만 이뤄졌는지, 소문자로만 이뤄졌는지 등을 확인

메서드	설명	사용 예
isalpha()	문자열이 숫자, 특수 문자, 공백이 아닌 문자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.isalpha()
isdigit()	문자열이 모두 숫자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.isdigit()
isalnum()	문자열이 특수 문자나 공백이 아닌 문자와 숫자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.isalnum()
isspace()	문자열이 모두 공백 문자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.isspace()
isupper()	문자열이 모두 로마자 대문자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.isupper()
islower()	문자열이 모두 로마자 소문자로 구성돼 있을 때만 True, 그 밖에는 False 반환	str.islower()

문자열 다루기

- 문자열의 구성 확인
  - 메서드 사용 예

```
In: print('Python'.isalpha())    # 문자열에 공백, 특수 문자, 숫자가 없음
    print('Ver. 3.x'.isalpha())  # 공백, 특수 문자, 숫자 중 하나가 있음
Out: True
     False
```

```
In: print('12345'.isdigit())      # 문자열이 모두 숫자로 구성됨
    print('12345abc'.isdigit())   # 문자열이 숫자로만 구성되지 않음
Out: True
     False
```

```
In: print('abc1234'.isalnum())    # 특수 문자나 공백이 아닌 문자와 숫자로 구성됨
    print('   abc1234'.isalnum())  # 문자열에 공백이 있음
Out: True
     False
```

```
In: print('   '.isspace())        # 문자열이 공백으로만 구성됨
    print(' 1 '.isspace())        # 문자열에 공백 외에 다른 문자가 있음
Out: True
     False
```

문자열 다루기

- 메서드 사용 예

```
In: print('PYTHON'.isupper())     # 문자열이 모두 대문자로 구성됨
    print('Python'.isupper())     # 문자열에 대문자와 소문자가 있음
    print('python'.islower())     # 문자열이 모두 소문자로 구성됨
    print('Python'.islower())     # 문자열에 대문자와 소문자가 있음
Out: True
     False
     True
     False
```

문자열 다루기

- 대소문자로 변경하기
  - lower()와 upper() 메서드 이용

```
str.lower()
str.upper()
```

  - lower()와 upper() 메서드 이용 예

```
In: string1 =  'Python is powerful. PYTHON IS EASY TO LEARN.'
    print(string1.lower())
    print(string1.upper())
Out: python is powerful. python is easy to learn.
     PYTHON IS POWERFUL. PYTHON IS EASY TO LEARN.
```

```
In: 'Python' == 'python'
Out: False
```

```
In: print('Python'.lower() == 'python'.lower())
    print('Python'.upper() == 'python'.upper())
Out: True
     True
```

텍스트 파일의 데이터를 읽고 처리하기

- 데이터 파일 준비 및 읽기
  - 데이터: 어느 커피 전문점에서 나흘 동안 기록한 메뉴별 커피 판매량
  - 작업: 나흘 동안 메뉴당 전체 판매량과 하루 평균 판매량 구하기

- 데이터 확인

```
In: !type c:₩myPyCode₩data₩coffeeShopSales.txt
Out: 날짜    에스프레소  아메리카노  카페라테  카푸치노
   10.15       10        50        45        20
   10.16       12        45        41        18
   10.17       11        53        32        25
   10.18       15        49        38        22
```

텍스트 파일의 데이터를 읽고 처리하기

- 데이터 읽기

```
In: # file_name = 'c:₩myPyCode₩data₩coffeeShopSales.txt'
    file_name = 'c:/myPyCode/data/coffeeShopSales.txt'
    f = open(file_name)          # 파일 열기
    for line in f:               # 한 줄씩 읽기
        print(line, end='')      # 한 줄씩 출력
    f.close()                    # 파일 닫기
Out: 날짜    에스프레소  아메리카노  카페라테  카푸치노
     10.15       10         50         45        20
     10.16       12         45         41        18
     10.17       11         53         32        25
     10.18       15         49         38        22
```

텍스트 파일의 데이터를 읽고 처리하기

- 문자열 데이터 처리

```
In: f = open(file_name)            # 파일 열기
    header = f.readline()          # 데이터의 첫 번째 줄을 읽음
    headerList = header.split()    # 첫 줄의 문자열을 분리한 후 리스트로 변환
    espresso = []                  # 커피 종류별로 빈 리스트 생성
    americano = []
    cafelatte = []
    cappucino = []
    for line in f:                 # 두 번째 줄부터 데이터를 읽어서 반복적으로 처리
      dataList = line.split()      # 문자열에서 공백을 제거해서 문자열 리스트로 변환
      # 커피 종류별로 정수로 변환한 후, 리스트의 항목으로 추가
      espresso.append(int(dataList[1]))
      americano.append(int(dataList[2]))
      cafelatte.append(int(dataList[3]))
      cappucino.append(int(dataList[4]))

    f.close() # 파일 닫기

    print("{0}: {1}".format(headerList[1], espresso)) # 변수에 할당된 값을 출력
    print("{0}: {1}".format(headerList[2], americano))
    print("{0}: {1}".format(headerList[3], cafelatte))
    print("{0}: {1}".format(headerList[4], cappucino))
Out: 에스프레소: [10, 12, 11, 15]
     아메리카노: [50, 45, 53, 49]
     카페라테: [45, 41, 32, 38]
     카푸치노: [20, 18, 25, 22]
```

텍스트 파일의 데이터를 읽고 처리하기

- 나흘간 메뉴별 전체 판매량과 하루 평균 판매량 계산

```
In: total_sum = [sum(espresso), sum(americano), sum(cafelatte), sum(cappucino)]
    total_mean = [sum(espresso)/len(espresso), sum(americano)/len(americano),
                  sum(cafelatte)/len(cafelatte), sum(cappucino)/len(cappucino) ]
    for k in range(len(total_sum)):
      print('[{0}] 판매량'.format(headerList[k+1]))
      print('- 나흘 전체: {0}, 하루 평균: {1}'.format(total_sum[k], total_mean[k]))

Out: [에스프레소] 판매량
     - 나흘 전체: 48, 하루 평균: 12.0
     [아메리카노] 판매량
     - 나흘 전체: 197, 하루 평균: 49.25
     [카페라테] 판매량
     - 나흘 전체: 156, 하루 평균: 39.0
     [카푸치노] 판매량
     - 나흘 전체: 85, 하루 평균: 21.25
```