CS101 – 텍스트 프로세싱

Lecture 20

School of Computing
KAIST

학습 목표:

- 대량의 자료를 포함하는 텍스트 파일을 만들어서 읽고 쓸 수 있다.
- 프로그램 구문 실행 흐름을 바꾸는 break와 continue를 활용할 수 있다.

파일

"planets.txt" 파일은 다음 내용을 가지고 있습니다.

```
Mercury
Venus
Earth
Mars
Jupiter
Saturn
Uranus
Neptune
```

```
>>> f = open("planets.txt", "r")
>>> s = f.readline()
>>> s, len(s)
('Mercury\n', 8)
```

f는 파일의 내용이 아니라 파일 객체입니다. (type: <class '_io.TextIOWrapper'>)
\n은 파일에서 읽어온 개행(줄 바꿈) 문자를 의미합니다.

파일에서 문자열 읽기

파일에서 읽어온 문자열 앞뒤의 공백 문자(줄 바꿈 문자, 띄어쓰기 등)를 제거하기 위해서 `strip()`, `rstrip()`함수를 사용합니다.
`print()`문에 `end=" "` 인자를 추가하면 문자열을 출력한 후 줄을 바꾸는 대신 한 칸을 띄어 쓸 수 있습니다.

```
>>> for l in f:
...     s = l.strip()
...     print(s, end=" ")
Venus Earth Mars Jupiter Saturn Uranus Neptune
```

파일 객체에 **for** 반복문을 사용하면 반복할 때마다 `readline()` 함수를 실행합니다.
반복문은 파일의 마지막 줄을 읽은 후에 종료됩니다.

파일 객체의 사용이 끝나면 `f.close()`함수를 실행해야 합니다.

파일 읽기

다음은 파일 전체의 내용을 읽어 리스트에 저장하는 프로그램입니다.

```
planets = []
f = open("planets.txt", "r")
for line in f:
  planets.append(line.strip())
f.close()
print(planets)
```

파일 객체는 위와 비슷한 일을 하는 멤버 함수를 가지고 있습니다.
(이 함수는 공백 문자를 따로 제거하지는 않습니다)

```
planets = f.readlines()
```

단어 찾기

파일에서 earth라는 단어가 포함된 줄의 위치를 찾고 싶습니다.

```
f = open("planets.txt", "r")
current = 0
earth = 0
for line in f:
  current += 1
  planet = line.strip().lower()
  if planet == "earth":
    earth = current
print("Earth is planet #%d" % earth)
```

이 프로그램은 earth 단어의 위치와 관계 없이 항상 파일 전체 내용을 읽지만,
단어를 찾은 이후에는 더 이상 파일을 읽을 필요가 없습니다.

빠른 단어 찾기

break 키워드는 현재 실행중인 반복문을 중지하고 빠져나옵니다.

```
f = open("planets.txt", "r")
earth = 0
for line in f:
  earth += 1
  planet = line.strip().lower()
  if planet == "earth":
    break
print("Earth is planet #%d" % earth)
```

break 를 사용하면 가장 안쪽의 반복문만 빠져나옵니다.

```
>>> for x in range(2):
...   for y in range(5):
...     print(y, end=" ")
...     if y == 3:
...       break
0 1 2 3 0 1 2 3
```

파일의 코멘트

파일의 내용에는 주석이 있을 수도 있습니다.
파일의 모든 주석이 #으로 시작한다고 하면, 다음과 같이 주석의 내용을 읽지 않고 건너뛸 수 있습니다.

```
f = open("planetsc.txt", "r")
earth = 0
for line in f:
  planet = line.strip().lower()
  if planet[0] == "#":
    continue
  earth += 1
  if planet == "earth":
    break
print("Earth is planet #%d" % earth)
```

continue 키워드는 현재 실행중인 반복을 건너뜁니다.

큰 파일

113,809개의 영어 단어로 이루어진 `words.txt` 파일의 자료를 사용해서 단어 게임을 해 봅시다 (https://en.wikipedia.org/wiki/Moby_Project)

단어의 길이가 18보다 긴 모든 영어 단어를 출력해 봅시다.
(예시: counterdemonstrations, 21글자)

```
f = open("words.txt", "r")
for line in f:
  word = line.strip()
  if len(word) > 18:
    print(word)
f.close()
```

단어 게임

글자 'e'가 포함되지 않은 단어의 수를 세 봅시다.

```
f = open("words.txt", "r")

count = 0
for line in f:
  word = line.strip()
  if not "e" in word:
    count += 1

print("%d words have no 'e'" % count)
f.close()
```

Abecedarian words

단어의 모든 글자가 알파벳 순서로 정렬된 단어들을 찾아봅시다.
(예시: art, allow, cello)

```
def is_abecedarian(word):
  for i in range(1, len(word)):
    if word[i-1] > word[i]:
      return False
  return True

f = open("words.txt", "r")

for line in f:
  word = line.strip()
  if is_abecedarian(word):
    print(word)

f.close()
```

연속된 동일한 문자 파악하기

동일한 두 글자가 연속해서 3번 이상 이어지는 단어는 얼마나 있을까요?
(예시: b**ookkee**per)
(유사하지만 해당되지 않는 경우: C**ommittee,** M**iss**i**ss**i**pp**i)

```
def three_doubles(word):
  s = ""
  for i in range(1, len(word)):
    if word[i-1] == word[i]:
      s = s + "*"
    else:
      s = s + " "
  return "* * *" in s
```

파일 쓰기

파일을 새로 만들고, 내용을 쓰는 것도 가능합니다.

```
f = open("./test.txt", "w")
f.write("CS101 is fantastic\n")
f.close()
```

파일을 쓰기 위해서는 `"w"` 모드를 사용해야 합니다.

파일 객체에는 파일에 내용을 쓰기 위한 `write(text)`멤버 함수가 있습니다.
`print()`와는 달리, `write()` 함수는 `text` 내용 출력 후 자동으로 줄을 바꾸거나 공백 문자를 넣지 않습니다. 줄을 바꾸고 싶다면 개행 문자 `'\n'`을 추가로 출력해야 합니다.

파일 사용이 끝나면 `close()`를 호출해야 합니다.
(호출하지 않으면 파일이 불완전하게 저장될 수 있습니다)

금융 자료

환율 자료를 이용해 실습을 해 봅시다. `1994.txt` ... `2009.txt` 파일은 해당 연도의 일별 KRW-USD 환율 정보를 가지고 있습니다. (www.oanda.com)

```
2009/05/11 0.00080110
2009/05/12 0.00083010
...
```

즉, 2009년 5월 11일에는 1 미국 달러가 약 1248.28 원이었습니다.
(1248.28 ≒ 1/ 0.00080110)

먼저, 모든 16개 파일을 읽어서, 각 라인의 문자열에 대해 `split` 함수를 사용하여, 날짜, 환율 정보가 담긴 튜플의 리스트로 만듭니다.

```
[... (20091227, 1154), (20091228, 1154),
(20091229, 1167), (20091230, 1167),
(20091231, 1163)]
```

최고, 최저 환율과 각 연도별 평균 환율을 구해봅시다.

```
Minimum: (19950705, 755)
Maximum: (19971223, 1960)
```

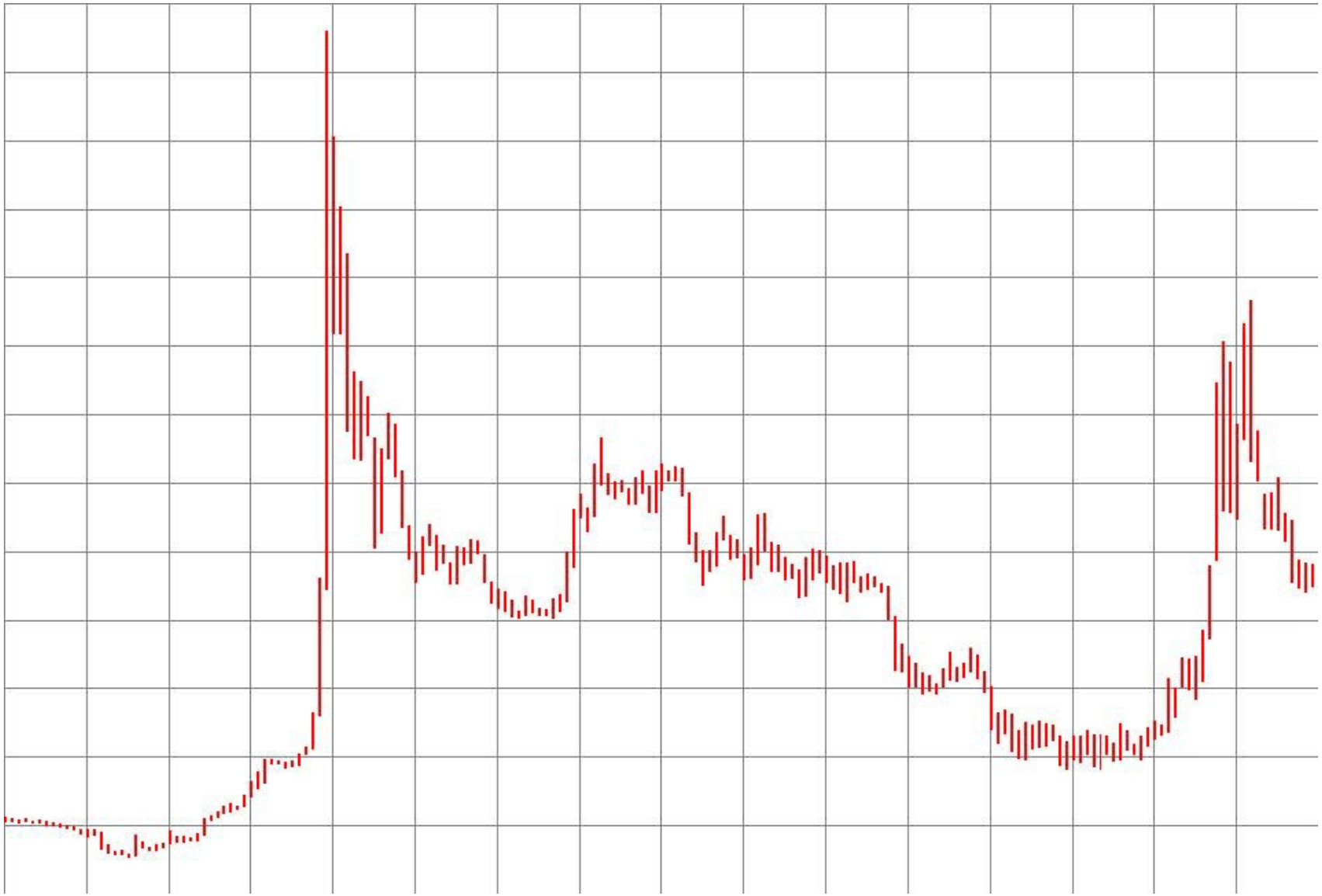
금융 자료

월별 최대, 최소 환율을 구해봅시다.

```
def find_minmax(yr):
  minmax = [ (9999, 0) ] * 12
  data = read_year(yr)
  for d, v in data:
    # make month 0 .. 11
    month = (d // 100) % 100 - 1
    minr, maxr = minmax[month]
    if v < minr:
      minr = v
    if v > maxr:
      maxr = v
    minmax[month] = minr, maxr
  return minmax
```

그래프 만들기

`cs1media` 모듈을 이용해 환율 변화 그래프를 만들어 봅시다.



정리 및 예습

본 강의 학습목표:

- 대량의 자료를 포함하는 텍스트 파일을 만들어서 읽고 쓸 수 있다.
- 프로그램 구문 실행 흐름을 바꾸는 break와 continue를 활용할 수 있다.

다음 강의 학습 목표:

- 블랙잭 카드 게임을 프로그램으로 만들기 위해 블랙잭 규칙을 이해할 수 있다.
- 블랙잭 카드 게임에 사용되는 카드를 객체로 표현할 수 있다.