

공학적 문제해결 기법

12주차 2차시

만들어져 있는 프로시저 사용하기2





12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

I. 개요

모듈 사용하기2

II. 학습 개요

1) 학습 목표

셸 자체를 컨트롤할 때 사용할 수 있는 시스템 함수들을 가지고 있는 sys모듈을 살펴보고, 시간을 가지고 작업하는 방법도 알아본다. 또한 pickle을 이용하여 정보를 저장하고 로드하는 방법도 학습한다.

2) 학습목차(세부목차)

- SYS 모듈로 셸 컨트롤하기
- TIME 모듈로 시간 작업하기
- 정보를 저장하기 위해 PICKLE 모듈을 사용하기



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 1 - SYS 모듈로 셸 컨트롤하기

sys 모듈 - stdin 객체

✓ 파이썬 셸 자체를 컨트롤할 때 사용할 수 있는 시스템 함수들을 가지고 있다.

✓ stdin과 stdout 객체, version 변수를 어떻게 사용하는지 살펴보자.

✓ stdin 객체로 읽기



stdin 객체 (standard input의 약자)는 사용자가 셸에 입력한 것을 읽음



엔터를 누를 때까지 키보드로 입력한 텍스트 라인을 읽는 readline 함수를 가지고 있다.



```
>>> import sys
```

```
>>> v = sys.stdin.readline()
```

```
He who laughs last thinks slowest
```

```
>>> print(v)
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 1 - SYS 모듈로 셸 컨트롤하기

sys 모듈 - stdin 객체

- ☒ input 함수와 readline 함수의 차이점들 중의 하나는 readline 함수는 매개변수로 지정된 글자 수만큼 읽어 들일 수 있다.

➡

```
>>> v = sys.stdin.readline(13)
He who laughs last thinks slowest
>>> print(v)
He who laughs
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 1 - SYS 모듈로 셸 컨트롤하기

sys 모듈 - stdout 객체

- ✅ stdout 객체 (standard output의 약자)는 셸 (또는 콘솔)에 메시지를 쓸 때 사용
- ✅ print 함수와 비슷하지만, stdout은 write와 같은 함수들을 가지고 있는 파일 객체



```
>>> import sys
```

```
>>> sys.stdout.write("What does a fish say when it swims into a  
wall?")
```

What does a fish say when it swims into a wall? 47

- ✅ write가 끝날 때, 쓰여진 글자 개수가 반환됨
- ✅ 화면에 얼마나 많은 글자를 썼는지 기록하기 위해서 이 값을 변수에 저장할 수도 있다.




12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 1 - SYS 모듈로 쉘 컨트롤하기

sys 모듈 - version 변수

- ☒ 내가 쓰는 파이썬 버전은?
- ☒ version 변수는 사용하고 있는 파이썬의 버전을 표시
- ☒ 최신 버전을 사용하고 있는지를 확인하고자 할 때 유용
- ☒ 어떤 프로그래머들은 그들이 만든 프로그램이 시작할 때 버전 정보를 보여주고 싶어하는데, 다음과 같이 프로그램에 파이썬 버전을 표시할 수 있다.



```
>>> import sys  
>>> print(sys.version)
```




12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간 작업하기

time 모듈

- ✓ time 모듈은 시간을 표시하는 함수들을 가지고 있다.

➡

```
>>> import time  
>>> print(time.time())  
1389269434.90625
```

- ✓ time()을 호출하여 반환된 숫자는 1970년 1월 1일 00시 00분 00초 이후 지금까지의 초를 나타냄
- ✓ 프로그램에서 실행하는데 걸리는 시간을 알아보고자 할 때 유용
- ✓ 0부터 999까지 모든 숫자를 출력하는데 걸리는 시간을 알아보도록 하자.
- ✓ 먼저 다음과 같은 함수가 필요하다.

➡

```
>>> def lots_of_numbers(max):  
    for x in range(0, max):  
        print(x)
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간 작업하기

time 모듈

- ☑ time 모듈을 가지고 이 함수가 시간이 얼마나 걸리는지 프로그램을 수정해 본다.

➡

```
>>> def lots_of_numbers(max):  
    t1 = time.time()  
    for x in range(0, max):  
        print(x)  
    t2 = time.time()  
    print('it took %s seconds' % (t2-t1))
```

991
992
993
994
995
997
998
999
It took 2.03125 seconds
>>>

- ☑ 0부터 999까지 모든 숫자를 출력하는데 걸리는 시간을 알아보도록 하자.
- ☑ 먼저 다음과 같은 함수가 필요하다.



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간작업하기

asctime으로 날짜 변환하기

- ✓ asctime 함수는 튜플로 날짜(date)를 받아서 읽을 수 있는 어떤 것으로 변환
- ✓ 튜플은 항목들을 가지고 있는 리스트와 같지만 변경할 수는 없음
- ✓ 아무런 매개변수 없이 asctime을 호출하면
- ✓ 읽을 수 있는 형태로 현재 날짜와 시간을 표시



```
>>> import time
```

```
>>> print(time.asctime())
```

```
Thu Jan 9 21:24:38 2014
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간작업하기

asctime으로 날짜 변환하기

- ✅ 매개변수로 asctime을 호출하려면 먼저 날짜와 시간에 대한 값으로 튜플을 생성
- ✅ 튜플을 변수 t에 할당

➡ >>> t = (2020, 2, 23, 10, 30, 48, 6, 0, 0)

- ✅ 이 값들은 년도와 월, 일, 시간, 분, 초, 요일(0은 월요일, 1은 화요일...), 일 년 중 며칠(0을 넣음), 일광 절약 시간인지 아닌지(만약에 아니라면 0, 맞다면 1)를 나타냄




12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간작업하기

asctime으로 날짜 변환하기

 튜플로 asctime을 호출하면 다음과 같은 값을 얻게 됨



```
>>> import time  
>>> t = (2020, 2, 23, 10, 30, 48, 6, 0, 0)  
>>> print(time.asctime(t))  
Sun Feb 23 10:30:48 2020
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간작업하기

localtime으로 날짜와 시간 얻기

- ✓ localtime 함수는 현재 날짜와 시간을 객체로 반환
- ✓ 이객체를 출력해보면 클래스이름과tm_year(년),tm_mon(월),tm_mday(한달중 며칠),tm_hour등의변수값을볼수있다.

➡

```
>>> import time
>>> print(time.localtime())
time.struct_time(tm_year=2014, tm_mon=1, tm_mday=9,
tm_hour=21, tm_min=32, tm_sec=19, tm_wday=3, tm_yday=9,
tm_isdst=0)
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간 작업하기

localtime으로 날짜와 시간 얻기

- ☒ 현재 연도와 월을 출력하려면 이들의 인덱스 위치를 사용해야 함
- ☒ year는 첫 번째 위치, month는 두 번째 위치 year=t[0], month =t[1]

➡

```
>>> t = time.localtime()
>>> year = t[0]
>>> month = t[1]
>>> print(year)
2014
>>> print(month)
1
```



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 2 - TIME 모듈로 시간 작업하기

sleep으로 잠깐 쉬기

✓ sleep 함수는 프로그램에서 약간의 딜레이를 주거나 천천히 동작하고자 할 때 유용하

✓ 1에서 60까지 1초씩 출력하고자 한다면?

➡

```
>>> for x in range(1, 61):  
    print(x)  
    time.sleep(1)
```

✓ 하나 출력하고 1초씩 쉬

✓ 즉, 각 숫자가 출력되는 사이에 딜레이(시간 지연)를 추가

✓ 애니메이션을 조금 더 그럴듯하게 보이도록 할 때 이 함수를 사용



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 3 - 정보를 저장하기 위해 PICKLE 모듈을 사용하기

pickle 모듈

- ✓ 객체를 파일로 쓸 수 있고 다시 쉽게 읽을 수 있는 어떤 것으로 변환해 주는데 사용
- ✓ 게임을 개발하고 있으며 플레이어의 진행 정보를 저장하고 싶다면 pickle이 유용
- ✓ 예를 들어 다음은 게임에 저장 기능을 추가한 것

➡ >>> game_data = {
 'player-position' : 'N23 E45',
 'pockets' : ['keys', 'pocket knife', 'polished stone'],
 'backpack' : ['rope', 'hammer', 'apple'],
 'money' : 158.50
}

- ✓ 가상의 게임에 있는 플레이어의 현재 위치와 플레이어의 주머니 및 가방에 있는 항목 목록, 그리고 가지고 있는 돈에 대한 정보를 가진 파이썬 맵을 생성



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 3 - 정보를 저장하기 위해 PICKLE 모듈을 사용하기

pickle 모듈

 이 맵을 파일로 저장하고 pickle의 dump 함수를 호출

➡

```
>>> import pickle
>>> game_data = {
    'player-position' : 'N23 E45',
    'pockets' : [ 'keys', 'pocket knife', 'polished stone' ],
    'backpack' : [ 'rope', 'hammer', 'apple' ],
    'money' : 158.50
}
>>> save_file = open('save.dat', 'wb')
>>> pickle.dump(game_data, save_file)
>>> save_file.close()
```

 매개변수로 wb로 save.dat 라는 파일을 연다. → 이 파일을 바이너리 모드로 쓰라고 하는 것



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 3 - 정보를 저장하기 위해 PICKLE 모듈을 사용하기

pickle 모듈

- ✓ 바이너리파일: 이미지와 음악파일, 영화, 파클된(pickled) 파이썬 객체들은 사람이 읽을 수 없는 정보를 가지고 있다.
- ✓ save.dat 파일을 열어보면? 텍스트 파일 같지 않은 것들이 보임

- ✓ pickle의 load 함수를 사용하여 파일로 쓴 객체를 언피클(unpickle)할 수 있다.
- ✓ 언피클은 피클(pickle) 작업을 역으로 하는 것



12주차 2차시 : 만들어져 있는 프로시저 사용하기 2

III. 학습 3 - 정보를 저장하기 위해 PICKLE 모듈을 사용하기

pickle 모듈

- ✓ 파일에 쓴 정보를 가져다가 프로그램이 사용할 수 있는 값으로 변환



```
>>> import pickle  
>>> load_file = open('save.dat', 'rb')  
>>> loaded_game_data = pickle.load(load_file)  
>>> load_file.close()
```

- ✓ 매개변수로 rb(read binary라는 뜻)를 사용해 파일을 열고
- ✓ 파일을 load 함수에 전달하고 반환값을 변수 loaded_game_data에 설정
- ✓ 마지막으로 그 파일을 닫는다.
- ✓ 저장된 데이터가 올바르게 로드되었는지 확인하기 위해 그 변수를 출력해본다.



```
>>> print(loaded_game_data)  
{'pockets': ['keys', 'pocket knife', 'polished stone'], 'player-  
position': 'N23 E45', 'money': 158.5, 'backpack': ['rope', 'hammer',  
'apple']}
```



『 이 콘텐츠는 2014학년도 학부교육 선도대학 육성사업(ACE)에 의하여 개발한 것임 』

