

공학적 문제해결 기법

7주차 2차시

빙글빙글
돌기2





7주차 2차시 : 빙글빙글 돌기2

I. 개요

복잡한 반복문 만들기

II. 학습 개요

1) 학습 목표

for문 안에 또 다른 for문을 중첩이라 하는데, 이러한 복잡한 루프에 대해 살펴보고, for루프와의 while루프도 알아본다. 이것은 서로 비슷하지만 다른 식으로 사용될 수 있다.

2) 학습 목차(세부 목차)

- 복잡한 for 루프
- while 루프



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

복잡한 for문

☑ 다음은 두 개의 코드 블록을 가진 복잡한 for 루프이다.

➡

```
hugehairypants = [ 'huge', 'hairy', 'pants' ]  
for i in hugehairypants:  
    print(i)  
    for j in hugehairypants:  
        print(j)
```

☑ hugehairypants라는 리스트를 생성한 다음

☑ 첫 번째 루프에 들어가서 리스트에 있는 하나의 항목을 출력

☑ 두 번째 루프에 들어가서 리스트에 있는 모든 항목을 출력

☑ 리스트의 다음 항목을 출력하는 print(i) 명령을 계속 진행

☑ 다시 한번 print(j) 로 리스트의 전체 항목을 출력



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

복잡한 for문

- ☒ 그 결과
- ☒ 붉은색 상자로 표시된 줄은 print(i) 문장에 의해 출력된 것
- ☒ 그 외 출력은 print(j)로 출력된 것

```
Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
huge
huge
hairy
pants
hairy
huge
hairy
pants
pants
huge
hairy
pants
>>>
```



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

복잡한 for문

앞서 다룬 예제에서 for를 이용하기

할아버지의 동전 복제 장치를 사용하여 일년 동안 얼마나 많은 금화를 얻을 수 있는지?

```
>>> 20 + 10 * 365 - 3 * 52
```

10개의 복제된 동전에 일년 365를 곱하고 발견한 20개의 동전을 더한 다음, 까마귀가 매주 훔쳐가는 동전 3개를 계산하여 뺀 것을 나타낸다.

매주 얼마의 금화가 더 생기는지 확인하기 위해 for 루프를 사용

매주 생기는 동전의 합계를 나타내는 magic_coins 변수의 값을 변경

magic_coins = 70 (매일 10개의 동전, 한 주는 7일)

coins 라는 변수를 생성하여 매주 증가하는 값을 확인



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

복잡한 for문



`found_coins = 20`

`magic_coins = 70`

`stolen_coins = 3`

① `coins = found_coins`

② `for week in range(1, 53):`

③ `coins = coins + magic_coins - stolen_coins`

④ `rint('Week %s = %s' % (week, coins))`



변수 `coins`에 `found_coins` 변수 값을 담음



루프가 돌 때 마다 변수 `week` 에는 1에서 52까지의 숫자가 차례로 담김



③은 기본적으로 매주마다 우리가 생성한 동전의 개수를 더하고 까마귀가 훔쳐가는 동전을 빼는 것



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

복잡한 for문

☒ ④는 print 구문으로, 몇 번째 주 인지와 지금까지의 동전 합계를 화면에 출력

☒ 실행결과

```
Python 3.3.3 Shell
File Edit Shell Debug Options Windows Help
Python 3.3.3 (v3.3.3:c3896275c0f6, Nov 18 2013, 21:18:40) [MSC v.1600 32 bit (Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Week 1 = 87
Week 2 = 154
Week 3 = 221
Week 4 = 288
Week 5 = 355
Week 6 = 422
Week 7 = 489
Week 8 = 556
Week 9 = 623
Week 10 = 690
Week 11 = 757
Week 12 = 824
Week 13 = 891
Week 14 = 958
Week 15 = 1025
```

```
Week 31 = 2097
Week 32 = 2164
Week 33 = 2231
Week 34 = 2298
Week 35 = 2365
Week 36 = 2432
Week 37 = 2499
Week 38 = 2566
Week 39 = 2633
Week 40 = 2700
Week 41 = 2767
Week 42 = 2834
Week 43 = 2901
Week 44 = 2968
Week 45 = 3035
Week 46 = 3102
Week 47 = 3169
Week 48 = 3236
Week 49 = 3303
Week 50 = 3370
Week 51 = 3437
Week 52 = 3504
>>>
```



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

for문 안에 또 다른 for문

- ☑ 예제로 많이 사용되는 구구단 출력 프로그램

➡ for x in range(2, 10):
 for y in range(1, 10):
 print ('%2d * %2d = %2d' % (x, y, x*y))
 print()

- ☑ 숫자자료형에 맞는 변환기호가 별도로 준비되어 있음
- ☑ %2d에서 %d는 정수형 변환기호이고
- ☑ %f는 실수형(소수점형) 변환기호
- ☑ %2d는 최소 2자리 범위를 두고 정수를 출력
- ☑ 위 출력에서 print ('%d * %d = %d' % (x, y, x*y))로 변경하여 출력해 보면 차이점을 알 수 있음



7주차 2차시 : 빙글빙글 돌기2

III. 학습 1 - 복잡한 for 루프

다양한 정수 출력 방법

- ☒ 정수를 출력하는 방법 중에서 10진(%d), 8진(%o), 16진(%x) 또는 16진 대문자 (%X)로 출력



```
>>> a=456
```

```
>>> print('%d -- %o -- %x -- %X' % (a, a, a, a))
```

```
456 -- 710 -- 1c8 -- 1C8
```



7주차 2차시 : 빙글빙글 돌기2

III. 학습 2 - while 루프

While문

- ☒ for 루프는 특정 길이만큼 반복
- ☒ 반복을 끝내야 할 시점을 알지 못할 경우에 사용
- ☒ 산길에 있는 계단을 상상해봅시다.
- ☒ 산이 정말로 높아서 정상에 오르기 전에 지칠 수 있으며 날씨가 갑자기 나빠져서 더 이상 오를 수 없게 될 수 있다.
- ☒ 이러한 경우에 while 루프 사용



7주차 2차시 : 빙글빙글 돌기2

III. 학습 2 - while 루프

While문

```
→ step = 0
while step < 10000:
    print(step)
    if tired == True :
        break
    elif badweather == True :
        break
    else :
        step = step + 1
```


- ☒ 입력하고 실행해보면 아직 tired 변수와 badweather 변수를 생성하지 않았기 때문에 에러가 날 것임




7주차 2차시 : 빙글빙글 돌기2

III. 학습 2 - while 루프


While 루프의 절차

-  1. 조건을 검사한다.
- 2. 블록에 있는 코드를 실행한다.
- 3. 반복한다

 while 루프는 하나의 조건문이 아닌 여러 개의 조건문을 가지고 생성

 `x = 45`
`y = 80`
`while x < 50 and y < 100:`
 `x = x + 1`
 `y = y + 1`
 `print(x, y)`

루프가 다섯 번 실행되면, 각 변수에 1씩 더

 루프가 다섯 번 실행되면, 각 변수에 1씩 더해지므로 변수 x는 50이 되고,
이는 첫 번째 조건(`x < 50`)에 맞지 않게 되어 루프를 종료



7주차 2차시 : 빙글빙글 돌기2

III. 학습 2 - while 루프

While문

- ☑ while 루프를 사용하는 또 다른 일반적인 예는 반영구적인 루프를 생성

➡ while True:

 코드 1

 코드 2

 코드 3

 if some_value == True:

 break

- ☑ while 루프의 조건이 True라는 것은 항상 참 → 그 블록안의 코드는 항상 실행

- ☑ 단, 변수 some_value가 참일 경우에는 루프를 빠져나올 수 있음

- ☑ 프로그램 짜보기

➡ 1부터 10까지의 합을 while 문을 이용하여 구하기



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

