

공학적 문제해결 기법

4주차 2차시

프로그램 짤 때 바탕이 되는 것 2





4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

I. 개요

이름 붙이기와 문자열

II. 학습 개요

1) 학습 목표

계산 물체에 이름을 붙이는 방법인 변수에 대해 알아보고, 값을 담기 위해 변수를 생성하고, 계산식에서 그 변수를 이용해 본다.

2) 학습 목차(세부 목차)





- 이름 붙이기
- 문자열 표현



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

계산 물체에 이름을 붙이기

-  프로그래밍 언어에서 아주 중요한 기능 가운데 하나
-  이름은 변수가 되고, 그 변수의 값은 계산 물체가 됨
-  변수(Variable)는 숫자와 문자, 숫자와 문자의 리스트 등의 정보를 저장하기 위한 장소를 의미
 - ☒ fred라는 변수를 생성하고 100이라는 값을 줌
 - ☒ `>>> fred = 100`
-  변수에 담긴 값을 확인하려면
 - ☒ `>>> print(fred)`
100



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

계산 물체에 이름을 붙이기

 fred 변수에 다른 값을 넣어 변수의 값을 변경

☒ >>> fred = 200

☒ >>> print(fred)
200








```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> fred=100
>>> print(fred)
100
>>> fred=200
>>> print(fred)
200
>>> |
```



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

계산 물체에 이름을 붙이기

-  동일한 값을 넣기 위해 하나 이상의 변수를 사용
-  `>>> fred = 200`
-  `>>> john = fred`
-  `>>> print(john)`
200
-  john이라는 이름의 변수가 fred와 같은 값을 갖길 원한다는 의미
-  Fred · 유용한 변수명이 아님
-  그 이름으로는 그 변수가 무엇을 위해 사용되는지 알 수 없기 때문



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

계산 물체에 이름을 붙이기

 fred라는 이름 대신에 `number_of_coins` 라는 변수를 사용

 `>>> number_of_coins = 200`

 `>>> print(number_of_coins)`
200







 이것은 200개의 동전에 대해 말하고 있음



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

Ⅲ. 학습 1 - 이름 붙이기

변수명

-  변수명은 문자와 숫자 그리고 밑줄(_)로 구성
-  숫자로 시작해서는 안 됨
-  한 문자(예를 들어 a)부터 긴 문장까지 무엇이든 가능
-  변수명에는 공백이 있어서는 안 됨
-  단어를 구분하기 위하여 밑줄 사용
-  빠르게 코딩하고 싶다면 짧은 변수명을 사용




4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2


III. 학습 1 - 이름 붙이기


변수 활용하기

앞에서, 지하실에 있던 할아버지의 멋진 기계로 새로운 동전을 만들 때 1년 후 얼마나 많은 동전들을 마술처럼 얻을 수 있는지에 대한 연산식을 기억하시나요?

 $\ggg 20 + 10 * 365$
3670

 $\ggg 3 * 52$
156

 $\ggg 3670 - 156$
3514

 이것을 한 줄의 코드로 변경 $\ggg 20 + 10 * 365 - 3 * 52$



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

변수 활용하기

앞에서, 지하실에 있던 할아버지의 멋진 기계로 새로운 동전을 만들 때 1년 후 얼마나 많은 동전들을 마술처럼 얻을 수 있는지에 대한 연산식을 기억하시나요?

 이 숫자들을 변수로 바꿔보자.

 >>> found_coins = 20

 >>> magic_coins = 10

 >>> stolen_coins = 3



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2


III. 학습 1 - 이름 붙이기

변수 활용하기

앞에서, 지하실에 있던 할아버지의 멋진 기계로 새로운 동전을 만들 때 1년 후 얼마나 많은 동전들을 마술처럼 얻을 수 있는지에 대한 연산식을 기억하시나요?

 `>>> found_coins + magic_coins * 365 - stolen_coins * 52`

 동일한 결과를 얻을 수 있다.

 변수는 값을 담는 방법이며, 나중에 사용할 수 있다는 점을 기억하자.



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 1 - 이름 붙이기

다시 계산을 하기 위해서 연산식 복사하기

1. 텍스트를 복사하기 위해 마우스로 선택한 다음 그 줄의 끝까지 드래그
2. CTRL 키를 누른 상태에서 C를 눌러 선택한 텍스트를 복사 (CTRL-C)
3. 마지막 프롬프트 클릭
4. CTRL 키를 누른 상태에서 V를 눌러서 선택한 텍스트를 붙여넣기 (CTRL-V)
5. 새로운 결과를 보기 위해서 Enter를 누름

```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> found_coins=20
>>> magic_coins=10
>>> stolen_coins=3
>>> found_coins+magic_coins*365-stolen_coins*52
3514
>>>
```

```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> found_coins=20
>>> magic_coins=10
>>> stolen_coins=3
>>> found_coins+magic_coins*365-stolen_coins*52
3514
>>> stolen_coins=2
>>> found_coins+magic_coins*365-stolen_coins*52
3566
>>> |
```



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

Ⅲ. 학습 1 - 이름 붙이기




여러분이 할아버지의 기계를 톡톡 쳐서 작동시킬 때마다
3개의 동전이 더 나온다면 1년에 몇 개의 동전을 얻게 될까?



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열

 프로그래밍 용어로 텍스트를 보통 문자열(string)이라고 부름

 모든 문자와 숫자, 기호들이 해당

 텍스트를 겹따옴표로 감싸서 문자열을 생성



```
>>> print("Hello")  
Hello
```



```
>>> fred = "Welcome to my lecture!!"
```



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2


III. 학습 2 - 문자열 표현

문자열

 `>>> print(fred)`
`Welcome to my lecture!!`

 홀따옴표를 사용하여 문자열을 생성

 `>>> fred = 'Welcome to my lecture!!'`

 `>>> print(fred)`
`Welcome to my lecture!!`



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열 에러 메시지

 >>> fred = "Welcome to my lecture!!"

 다음과 같은 결과를 보게 됨

 SyntaxError: EOL while scanning string literal

```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> fred = "Welcome to my lecture!!"
>>> print(fred)
Welcome to my lecture!!
>>> fred = 'Welcome to my lecture!!!'
>>> print(fred)
Welcome to my lecture!!!
>>> fred = "Welcome to my lecture!!"
SyntaxError: EOL while scanning string literal
>>>
```

 구문에 대한 에러 메시지

 끝나는 문자열에 홑따옴표나 겹따옴표를 사용해야 하는 규칙을 어김



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열 에러 메시지

 구문(Syntax) : 문장 안에서 단어들의 배치와 순서를 의미

 `SyntaxError`

- ☒ 예상하지 못한 순서로 되어 있다는 뜻
- ☒ 파이썬이 기대한 무언가를 빠뜨렸다는 의미

 `EOL`




- ☒ `end-of-line`



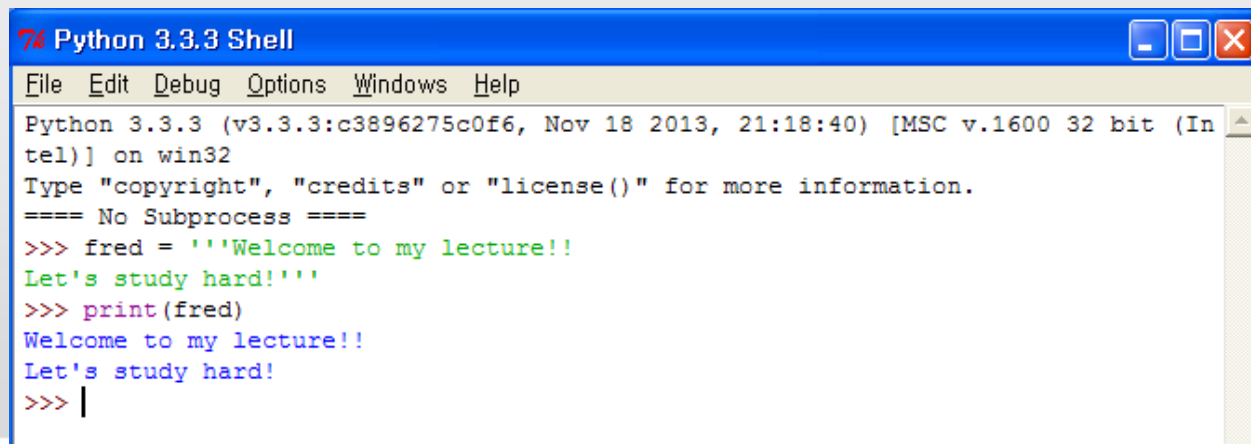
4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

멀티라인 문자열

-  한 줄 이상의 여러 줄로 된 텍스트를 사용
-  세 개의 홑따표('')를 사용하고, 각 줄의 사이에 Enter를 입력
- 

```
>>> fred = '''Welcome to my lecture!!
Let's study hard!'''
```










```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> fred = '''Welcome to my lecture!!
Let's study hard!'''
>>> print(fred)
Welcome to my lecture!!
Let's study hard!
>>> |
```



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

멀티라인 문자열

-  세 개의 홑따옴표를 사용한다면 문자열 내에 홑따옴표와 겹따옴표의 조합해서 사용가능
-  `silly_string = 'He said, "Aren't can't shouldn't wouldn't."'`
-  홑따옴표나 겹따옴표를 정말로 사용하고 싶다 · 문자열에 있는 각각의 따옴표 앞에 백슬래시(\)를 추가
-  이것을 이스케이핑(escaping)이라 함
-  `>>> single_quote_str = 'He said, "Aren\\'t can\\'t shouldn\\'t wouldn\\'t."'`
-  `>>> double_quote_str = "He said, \\\"Aren't can't shouldn't wouldn't.\\\""`
-  백슬래시 문자는 출력할 때 나타나지 않는다




4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열에 값을 삽입하기

 변수에 값을 이용하여 메시지를 표시 → %s를 이용

 >>> myscore = 1000

 >>> message = 'I scored %s points'

 >>> print(message % myscore)
I scored 1000 points

 print(message % 1000)라고 해도 동일한 결과



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열에 값을 삽입하기

```
Python 3.3.3 Shell
File Edit 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.
==== No Subprocess ====
>>> myscore = 1000
>>> message = 'I scored %s points'
>>> print(message %myscore)
I scored 1000 points
>>> print(message % 1000)
I scored 1000 points
>>>
```



4주차 2차시 : 프로그램 짤 때 바탕이 되는 것 2

III. 학습 2 - 문자열 표현

문자열에 값을 삽입하기

다음예를 실행해보기

```
>>> furniture = '%s: a device for finding furniture in the dark'
```

```
>>> bodypart1 = 'Knee'
```

```
>>> bodypart2 = 'Shin'
```

```
>>> print(furniture % bodypart1)
```

```
>>> print(furniture % bodypart2)
```

```
>>> nums = 'What did the number %s say to the number %s?'
```

```
>>> print(nums % (0, 8))
```




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

