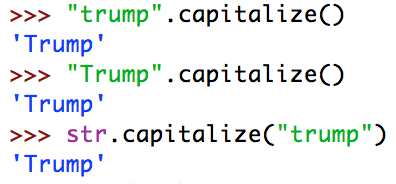
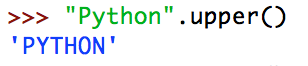
Week 6

* str 매서드

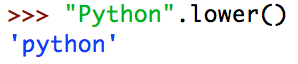
* + capitalize() :  텍스트 맨 첫 글자를 대문자로 바꾸어준다.(대문자면 변화 x) 출력값 type 은 str



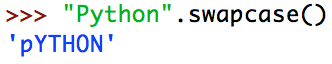
* + upper() : 문자열 중 소문자를 모두 대문자로 바꾼다.(대문자면 변화 x) 출력값 str



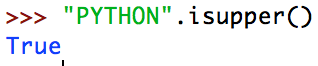
* + lower() : 문자열 중 대문자를 모두 소문자로 바꾼다.(소문자면 변화 x) 출력값 str



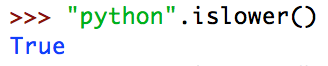
* + swapcase() : 대문자면 소문자로 소문자면 대문자로 바꾼다. 출력값 str



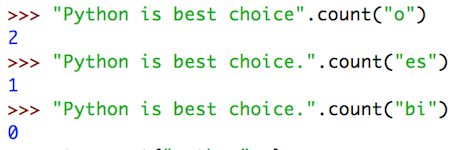
* + isupper() : 문자열이 모두 대문자이면 True를 출력한다. 출력값  bool

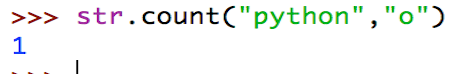


* + islower() : 문자열이 모두 소문자이면 True를 출력한다. 출력값 bool

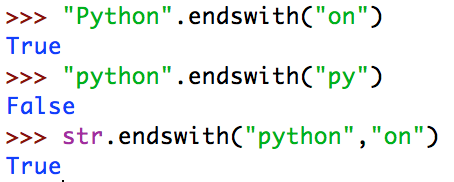


* + count("찾을문자") : 찾을 문자(굳이 한글자가 아니어도 된다.)의 개수를 리턴한다. 출력값은 int





* + startswith("내용") : 해당 문자열이 "내용"으로 입력한 것으로 시작하는지 확인하는 함수 출력값은 bool(True, False)
  + endswith("내용") : 해당 문자열이 "내용"으로 입력한 것으로 끝나는지 확인하는 함수 출력값은  bool(True , False)

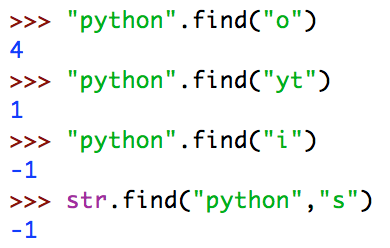


* + find("내용", 시작번호, 끝번호) :  문자열중 "내용"이 처음 나온 위치(파이썬은 처음이 0부터 시작)를 반환한다.

시작번호 ,끝번호는 생략할 수있다. 시작번호 생략하면 0으로 시작,  시작번호를 쓰고 끝번호 생략하면 맨 끝까지 찾는다.

만약 문자열중 "내용"이 존재 않는다면 -1을 반환한다.

출력값은 int

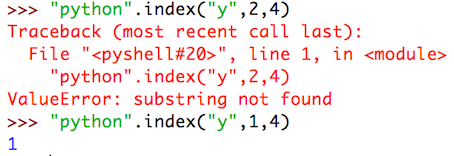
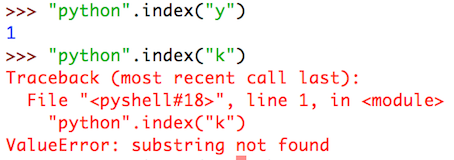


* + index('내용',시작번호,끝번호) : find와 마찬가지로 문자열중 "내용"이 처음 나온 위치(파이썬은 처음이 0부터 시작)를 반환한다.

시작번호 ,끝번호는 생략할 수있다. 시작번호 생략하면 0으로 시작,  시작번호를 쓰고 끝번호 생략하면 맨 끝까지 찾는다.

find와 다른 점은 문자열 중 "내용"값이 존재하지 않으면 오류(ValueError)가 난다.

출력값은 int



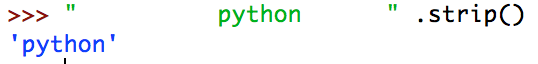
* + lstrip() : 문자열 중 가장 왼쪽에 있는 한 칸 이상의 공백을 모두 지운다. 출력값 str



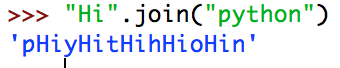
* + rstrip() : 문자열 중 가장 오른쪽에 있는 한 칸 이상의 공백을 모두 지운다. 출력값 str

https://khlug.org/image/6435ad00e7770167.png

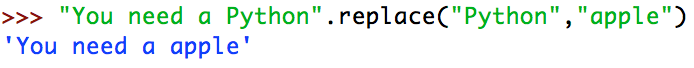
* + strip() : 문자열 중 양 쪽에 있는 한 칸 이상의 공백을 모두 지운다. 출력값 str



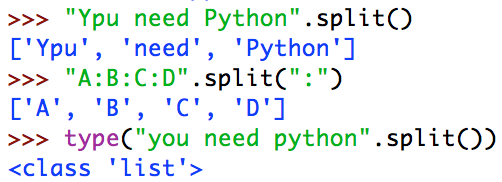
* + join("내용") : "내용"안의 각각의 문자 사이에 문자열을 집어넣는다. 헷갈릴 수도 있는데 함수 앞에 나오는 문자가 삽입되는 문자열이다. 출력값은 str



* + replace("기존의 것","새로운 것") : 기존에 있던 문자열을 새로운 문자열로 바꾸는 함수이다. 출력값 str



* + split("내용") : "내용"값을 기준으로 문자열을 나누어 준다. 만약 "내용"값이 없으면 공백을 기준으로 나눈다.  나눈 항목들은 리스트의 요소로 저장이 된다. 출력값  list



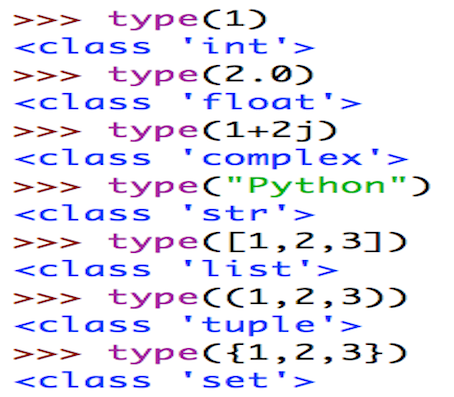
.format함수는 문자열 포맷팅 공부 할 때 같이 써야겠다

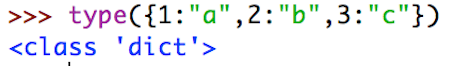
* 박정진#37887

2018.04.11. 16:46:38 (145일 전)   
수정 2018.04.13. 10:14:22 (143일 전)

[이동](https://khlug.org/group/35951) [수정](https://khlug.org/group/35951) [삭제](https://khlug.org/group/35951) [보관](https://khlug.org/group/35951)

Week 2에서 자료의 형태 type 을 배웠는데 지금까지 안  type을 정리해보면





type함수를 써서 각각의 객체의 타입을 확인해보면 출력값에 <class  'type'> 이라고 나온다.

class는 파이썬이 type을 보여주는 방법이다.

각각의 기본으로 파이썬이 갖고있는 class들은 그 클래스가 갖고있는 고유의 method(매서드)가 있다.

그리고 우리가 클래스를 만들 수 있고 상속 부터 여러가지 기능을 배우는데 그것은 Week 9내용이므로 나중에 해야겠다.

모듈이랑 다른 점은 모듈은 모듈안에 있는 변수, 함수를 부를  때

* + 모듈이름.모듈함수(함수를 적용할 인자)
  + 모듈이름.모듈변수

이렇게 .을 이용하여 불렀다. .앞에 모듈이름이 들어가야만 했었다.

클래스는 조금 다르다. 클래스도 똑같이 .을 이용하여 함수(매서드)를 부르는데 두가지 방법이 있다.

* + 해당 클래스 이름. 함수(매서드)이름(함수를 적용할 인자)
  + 함수를 적용할 인자.함수(매서드)이름()
  + 인자를 할당한 변수.함수(매서드)이름()

이렇게 . 앞에 바로 객체를 불러서 사용을 하는 점 때문에 객체 지향적인 방법으로 매서드를 부른다고 한다.

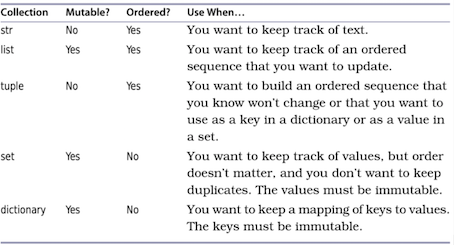
함수를 적용할 인자는 그 인자의 클래스에 있는 매서드를 써야된다.

빈 괄호()안에는 그 함수에 필요한 다른 인자들이 들어갈 수도 있고 생략해서 빈 괄호()로 쓸 수 도 있다.

위에 보여진 알고있는 모든 클래스에는 그 클래스가 갖고 있는 매서드가 있다.

Week 5에서는 문자열 , 리스트를 배운다. 둘 다 반복가능한 자료형(시퀸스)이므로 매우 중요한 타입이다.

문자열 매서드를 배우기 전에 시퀸스 종류에 대해 정리해보면



반복 가능한 자료형(시퀸스)를 변할수 있는지 와 순서가 정해져있는지 에 대한 특징으로 구별 해놓은 표이다.(Week 8 Data collection type)

정리를 하면

|  |  |  |  |
| --- | --- | --- | --- |
| 타입 | 변할 수 있는지 ? | 순서가 정해져 있는지? | 언제 쓰는지? |
| str(문자열) | No | Yes | 텍스트를 조사(추적)하고 싶을 때 |
| List(리스트) | Yes | Yes | 업데이트(변경)이 가능한 순서가 있는 자료를 원할 때 |
| tuple(튜플) | No | Yes | 변경이 불가능한 순서가 있는 자료를 원할 때 혹은 딕셔너리에서 key로 사용하려거나 혹은 set의 value(값)으로 사용하고자 할 때 |
| set(집합) | Yes | No | 순서는 상관없는 값(value)만 있고 복제가 불가능한 자료를 원할 때 (값은 바뀌지 않는다) |
| Dictionary(딕셔너리) | Yes | No | 키(key)에 대응하는 값(valule)을 찾고 싶을 때 (value값은 바뀌지 않는다.) |

str매서드가 뭐가 있는 지 보고싶으면

help(str)을 입력하면 되는데 처음에 \_\_add\_\_ 등 처음 본 것들이 나올 것인데 str이라는 클래스는 그위에 object(객체)라는 클래스에서

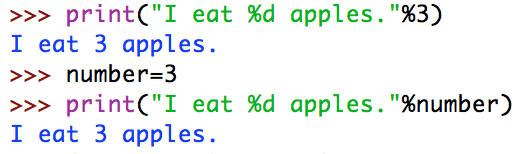
상속(inheritance)받은 클래스이므로 object가 가진 매서드도 갖고있는 것이다.

따라서 일단 capitalize함수 밑에서 부터 보면 될 듯 하다.

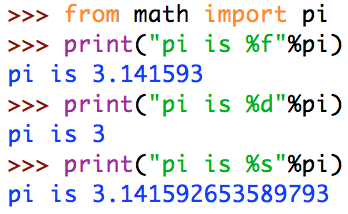
우리는 지금까지 print 할 때 +, \*,  , (세미콜론), sep= , end=  등을 이용해서 문자열을 만들어 출력했다.

하지만 문자열 포매팅을 공부하면 그럴 필요가 없다.

먼저 기본적으로 %을 이용해 포맷팅을 하는 것이다.



위에서 보는 것 과 같이 %d부분에 따옴표가 끝나고 넣은 %내용 부분이 들어가 출력되는 것을 볼 수있다. %에 들어가는건 자료여도 되고 변수여도 된다.



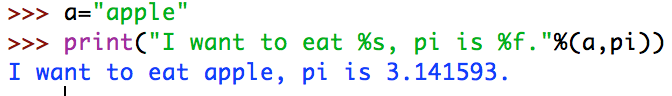
math함수를 이용해 pi를 불러보고 %d %s %f를 해보자.

%d는 정수형 포맷 코드여서 3이 출력되는것을 확인 할 수있고

%f는 float포맷 코드여서 실수가 출력되는 것을 확인(부동 소수여서 표시하는 값에 제한이 있다)

%s는 문자열 포맷 코드여서 pi가 문자열로 바뀌어 출력되는 것을 확인

물론 두개 이상 포맷팅도 가능하다.



|  |  |
| --- | --- |
| 코드 | 설명 |
| %s | 문자열 |
| %d | 정수형 |
| %f | 실수형 |
| %c | 문자 한 개 |
| %o | 8진수 |
| %x | 16진수 |
| %% | %를 출력 |

각 자료형을 포맷팅할 때 집어넣는 값은 각 자료형에 맞게 끔 설정 안해도된다. 즉 실수를 넣고 %d로 포맷팅을하면 정수형으로 출력되기 때문에 자료형의 형태는 상관없다.

중요한건 %s인데 %s 는 어떤 형태이든 변환해 넣을 수 있다. 따라서 %s 를 쓸 땐 상관 없이 쓰면 된다.

포맷코드랑 숫자도 함게 사용할 수 있다.(do it  파이썬)

1.정렬과 공백

%숫자형태(ex %10s)로 정렬과 공백을 할 수있다.

절대값(숫자) : 전체 칸

양수 : 공백 먼저 문자 오른쪽에 정렬(전체칸- 문자칸= 공백칸)

음수 : 문자 먼저 왼쪽에 정렬 공백  채워짐(전체칸-문자칸=공백칸)

만약 "%10s" %"hi" 면 hi가 오른쪽에 정렬되고 앞에는 8칸 공백이 있는 것이다.

2. 소수점 표현하기

%x.yf(%f)

전체 x칸 y번째 자리 까지 표시

x에 관해서 는 위와 똑같다.

절대값(숫자) : 전체 칸

양수 : 공백 먼저 문자 오른쪽에 정렬(전체칸- 문자칸= 공백칸)

음수 : 문자 먼저 왼쪽에 정렬 공백  채워짐(전체칸-문자칸=공백칸)

* .format을 이용한 문자열 포매팅

%로 포맷팅 하는것과 비슷한데 %s 처럼 모든 것을 포매팅 할 수있지만 자료형은 Str으로 바꿔주지 않고 그대로 넣어주는 기능이 있다.

%로 포매팅 하는 것과 같이  하면 된다. 변수를 넣어도 되고 값으로 넣어도되고  혼용해서 넣어도 되고 상관없다. 인자가 들어가는 순서만 지키면 제대로 포매팅이 된다.

파이썬은 0부터 시작하기 때문에 {0},{1},{2}.... 순서로 가고 맨 뒤에 .format(x,y,z,....)형태로 넣으면 0,1,2 순서로 x,y,z  가 들어간다.

순서말고 {변수} ,{ 값}으로 들어가도 상관없고 또한 {}으로 쭉 채워도 된다. 하지만 항상 지켜지는 규칙은 .format다음에 들어가는 인자 순서대로 배정이 된다는 것이다.

format을 이용하여 %처럼 정렬 및 소수점 표현을 할 수 도있고 공백을 채울 수 있는 방법도 있다.

1.정렬 및 공백

왼쪽 정렬 {순서:< x} : {순서}에 전체 x칸으로 문자 먼저 왼쪽 정렬을 하고 나머지 칸에 공백이 채워짐.(전체칸-문자칸=공백칸)

오른쪽 정렬 { 순서:>x} : {순서}에 전체 x칸으로 공백 먼저 문자 오른쪽 정렬을 한다. (전체칸-문자칸=공백칸)

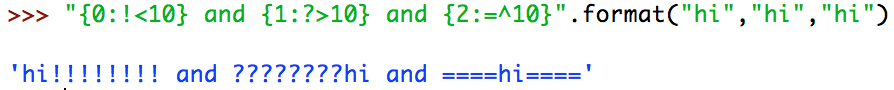
가운데 정렬 {순서:^x} : {순서}에 전체 x칸으로 문자열을 제외하고 왼쪽 오른쪽 균일하게 공백이 생긴다.(전체칸-문자칸=공백칸)

공백 채우기 {순서:내용<x} : { 순서 }에 전체 x칸으로 문자 왼쪽 정렬하고 나머지 공백에 내용이 채워진다.

{순서:내용>x} : {순서}에 전체 x칸으로 문자 오른쪽 정렬하고 나머지 공백에 내용이 채워진다.

{순서:내용^x} : {순서}에 전체 x칸으로 문자를 가운데 정렬하고 나머지 공백에 내용이 채워진다.

(위에 정렬에 : 와 >, <, ^ 기호 사이에 공백을 채울 내용을 넣으면 된다.)

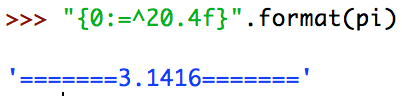


2. 소수점 표현하기

{순서:x.yf} :f는 float

%와 마찬가지로 전체 x칸으로 소수  y번째까지 표시를 해준다. 아무것도 안쓰면 자동으로 오른쪽  정렬이되고 위에 처럼 >, < , ^을 :과 x 사이에 넣으면 왼쪽정렬, 가운데 정렬을 할 수있고

또한 공백채우는 것도 방법은 위와 같다.



3.{   } 표시

{{    }} 처럼 그냥 중괄호({})에 {}을 씌우면 { } 을 출력할 수 있다.

 list

list  도 마찬가지로 list만의 매서드가 있다.

리스트의 성질.

1. 리스트는 기본적으로 [ ]으로 표현한다.
2. 리스트는 하나의 타입이다.
3. 리스트는 객체이다. (객체는 변수에 할당 될 수있으므로 변수=리스트 형태로 변수에 리스트를 할당할 수있다.)
4. 리스트 안의 요소(element)들이 있다.
5. []는  요소가 없는 리스트고 bool로 나타냈을 땐 False값이다.
6. 리스트는 객체여서 하나의 메모리 주소를 갖고있는데 그 리스트 안에서 요소들(객체)들의 메모리 주소도 포함하고 있다.
7. 리스트의 요소는 어떤 자료형이든 들어갈 수있다.(정수, 실수, 문자형, 리스트, 튜플 등등)(하지만 보통 같은 자료형을 담고 있다.)
8. 리스트의 요소는 변경이 가능하다. (문자열은 불가능하다)
9. 리스트를 인덱싱할 때 다른 자료형과 마찬가지로 처음은 0이다.

리스트의 인덱싱과 슬라이싱은 Week 3에 문자열 인덱싱 슬라이싱이랑 개념을 같이한다. 문자열 인덱싱 슬라이싱 했던 것 처럼 그대로 하면 된다.

추가로 리스트는 중첩이 될 수 있다. 리스트의 요소가 리스트인 경우 슬라이싱을 어떻게 하냐면

a[i][j] 처럼 뒤에 계속 [번호]를 붙여주면 리스트의 리스트의 리스트도 인덱싱 슬라이싱을 할 수있다.

a[1][2]면 a의 두번 째 요소의 리스트의 세번 째 요소를 인덱싱 하는 것이다.

이중, 삼중, 사중 계속 가능하다.

리스트는 mutable한 속성에 대해서 알아보면 리스트는 기본적으로 수정, 삭제를 할 수 있다. 첨가도 가능한데 첨가하는 방법은 많으므로 리스트 매서드 할 때 정리해야겠다.

 하나씩 정리해 보면

* 수정

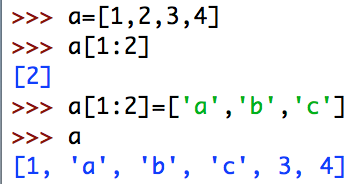
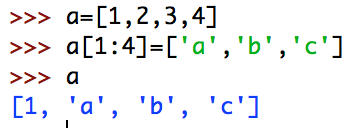
1. 리스트에서 하나의 값 수정하기

L[i]=x

리스트의 i번 째 항목이 x로 수정된다.

주의 : x=L[i]로 입력하면 변수 x에 리스트의 i번째 항목(객체)가 할당된다.

2. 연속된 범위의 값도 수정이 가능하다.

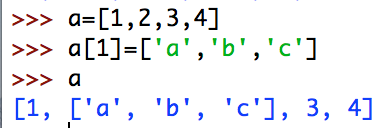
  

이렇게 [1:2] 라고 1부터 2전까지 라고 범위를 지정해두면 1은 포함 2는 포함하지않으므로 a[1]이라는 값을 슬라이싱해서 보여주지만

a[1:2]는  연속된 값으로 처리되므로 문자열 'a','b','c'가 2 대신에 들어 간 것을 확인 할 수 있고 (a[1]과 a[2]사이의 리스트를 ['a','b','c']로 수정)

오른쪽 예에서도 볼 수있듯이 a[1:4]는 1부터 3까지 범위 즉 2,3,4가 'a','b','c'로 바뀐 것을 확인 할 수 있다.(a[1]과 a[3](포함)사이의 리스트를 ['a','b','c']로 수정)

주의 :

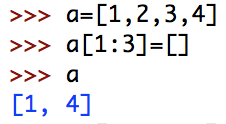


이처럼 한 요소만 저렇게 리스트로 바꾸면 리스트 안의 a[1]의 요소가 리스트 자체가 되버린다.

a[1]의 요소를 ['a','b','c']로 수정한다는 뜻이므로 전혀 다른 결과 값이 나오는 것을 확인 할 수있다.

* 삭제

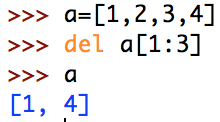
1.[]을 이용해 삭제하기



a[1:3]을 빈 공백으로 수정하라는 뜻인데 a[1], a[2]를 빈 공백으로 수정하니까 삭제 된 것을 확인 할 수있다.

하나의 값, 연속된 범위 둘 다 적용된다.

2. del 함수를 이용하여 삭제하기



del함수는 파이썬이 자체적으로 갖고있는 내장함수이다. 객체를 삭제하는 함수인데

위의 예에서는 a[1:3]을 삭제하라고 했으니까 a[1], a[2]가 삭제 된 것을 확인 할 수있다.

하나의 값, 연속된 범위 둘 다 적용된다.

리스트의 연산

len(L)(리스트 요소의 개수)

max(L)(리스트 중 최대값)

 min(L)(리스트 중 최소값)

sum(L)(리스트 요소의 합)

 sorted(L)(작은 것부터 큰 것 순으로 정렬)

이 5개가 가능하다. ( 함수를 쓸 때 입력값은 리스트이다.)

리스트 연산자

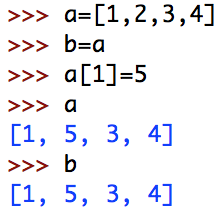
리스트는 +와 \*으로도 연산이 가능하다

+ : 두개의 리스트를 하나의 리스트로 합친다

\* : 한개의 리스트를 곱한 만 큼 반복한다.

Aliasing vs Copying

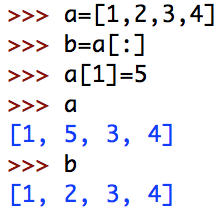
Aliasing : 두개의 변수가 하나의 리스트(객체)를 가르켜 한 변수의 리스트가 수정되면 다른 변수의 리스트도 수정이 된다.



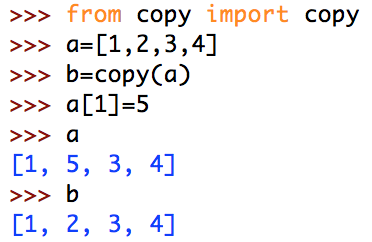
Copying : 기존의 리스트에서 새로운 리스트를 만들어 그 리스트가 기존의 리스트 요소(객체)를 가르켜  리스트가 수정되어도 새로운 리스트는 수정되지 않는다.

copying 하는 방법은 두가지 이다.

1.[:] (리스트 전체)를 이용하는 방법



2.copy 모듈을 이용하는 방법

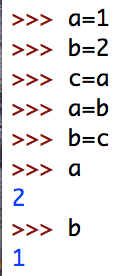


변수가 같은 객체를 가리키고 있는지 확인 하는 방법으로는 파이썬 내장 함수로 있는 is 를 쓰면 된다.

a is b 를 했을 때 True면 같은 객체를 가리키고 있는 것이고 False면 다른 객체를 가리키고 있는 것이다.

Tip : 변수를 바꾸는 방법

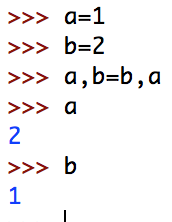
1. 기존 방법



a=1 ,b=2라고 설정하고

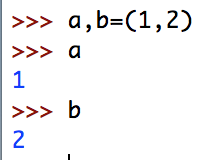
c라는 변수에 a의 값을 담아두고 a변수가 b변수 객체를 가르키게 하고 b변수는 기존 a변수 객체를 담아둔 c를 가리키게 해  a,b값을 서로 바꾸었다.

2.파이썬에서 간단하게 처리하기



그냥 a,b=b,a를 하면 서로의 객체가 바뀐다. 정말 파이썬의 간단함을 보여주는 것 같다.

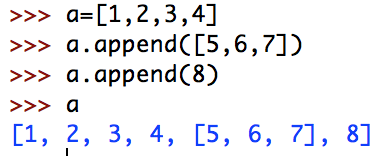
Tip : Week 2에서 함수를 다룰 때 잠깐 설명했던 것 처럼 한 줄에 여러개의 변수를 만들어 각각의 변수에 다른 객체를 할당 하고 싶으면 튜플 형태로 지정하면 된다.



list 매서드

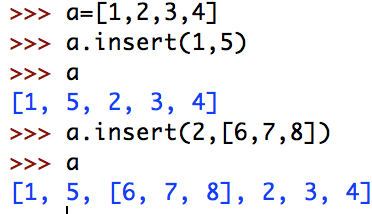
* 추가

* append(x) : 리스트의 맨 마지막 요소에 x 를 추가한다. (x는 리스트에 들어갈 수있는 어떠한 값이어도 상관없다.)(append함수는 요소 하나만 추가하는 함수)

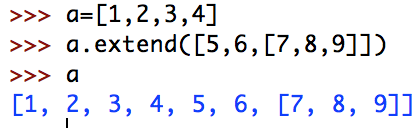


a에 리스트 요소가 추가 되고 그냥 정수형 도 추가 된 것을 확인 할 수있다.(추가되는 위치는 항상 맨 마지막에 추가된다)

* insert(x,y) : 리스트의 x번째 위치에 y를 추가하는 함수 (append와 마찬가지로 요소 하나를 추가하는 것이지만 다른 점은 추가할 위치를 지정할 수 있다는 것이다.)

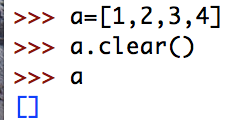


* extend(x) : 리스트의 맨 마지막에 리스트 x를 추가 하는 함수(a=a+[...] 와 같은 함수 이다.) extend로 리스트 요소를 추가하고 싶으면 [[...]]로 두번 씌우면 된다.

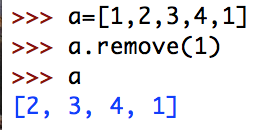


* 삭제

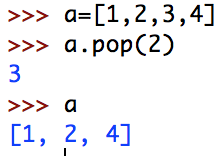
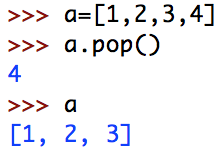
* clear() : 리스트에 있는 모든 요소를 삭제하는 함수



* remove(x) : 리스트에서 첫 번째로 나오는 x를 삭제하는 함수

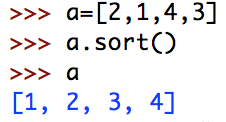


* pop(x)  : x(인덱싱넘버)는 생략가능하고 생략하면 리스트의 맨 마지막 요소를 출력하고 그 요소는 삭제한다. x에 값을 넣으면 x번째 요소를 출력하고 그 요소는 삭제한다.

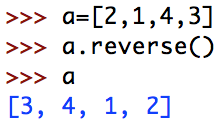


* 정렬

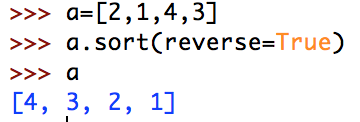
* sort() : 숫자가 작은 순부터 큰 순으로 , 알파벳 순서대로 리스트의 요소를 정렬한다.



* reverse() : 현재 리스트의 요소를 역으로 정렬한다. (sort한 상태에서 역정렬이 되는 것이 아니라 기존의 정렬이 역정렬이 되는 것임)



* sort(reverse=True) : sort와는 역방향으로  정렬한다.(sort()를 하고 reverse()를 쓴 상태와 같다)



* 그 외

* count(x) : 리스트의 x의 개수를 리턴한다 출력값 int

* index(x,시작번호,끝번호) : str매서드의 index와 똑같은 기능을 한다.

       리스트 요소 중 x가 처음 나온 위치(파이썬은 처음이 0부터 시작)를 반환한다.

       시작번호 ,끝번호는 생략할 수있다. 시작번호 생략하면 0으로 시작,  시작번호를 쓰고 끝번호 생략하면 맨 끝까지 찾는다.

      리스트 요소 중 x값이 존재하지 않으면 오류(ValueError)가 난다.

      출력값은 int