

# 5주차 데이터 수집 (크롤링)

데이터를 수집하는 방법은 여러가지가 있습니다. csv(comma seperated values)나 엑셀과 같이 이미 정리된 파일 형식으로 데이터를 얻을 수도 있고, Database나 API를 통해 얻을 수도 있지만, 오늘은 웹 크롤링을 통해 데이터를 수집하는 방법에 대해서 배워보겠습니다. Web crawling은 Web scraping라고도 말하는데, 위키피디아에 따르면 '조직적이고 자동화된 방법으로 월드 와이드 웹을 탐색하는 방법'이라고 나와있네요. HTML 파일을 긁어서 필요한 정보들만 뽑아 csv 파일로 저장하는 방법에 대해 배워보겠습니다.

웹사이트 크롤링을 하기 위해선 우선 웹사이트를 구성하고 있는 **HTML, CSS**에 대한 기본적인 지식이 필요합니다.

## 오늘 수업에 필요한 패키지를 설치해주세요

```
> pip install beautifulsoup4  
> pip install requests  
> pip install numpy  
> pip install pandas
```

개발 도중 사용법에 대해 궁금한 것들이 있으면 Document를 참고해보세요.

- [beautifulsoup4 \(<http://coreapython.hosting.paran.com/etc/beautifulsoup4.html>\)](http://coreapython.hosting.paran.com/etc/beautifulsoup4.html): HTML로부터 데이터를 뽑아내기 위한 라이브러리입니다.
- [requests \(<http://docs.python-requests.org/en/master/>\)](http://docs.python-requests.org/en/master/): requests.get을 통해 HTML을 받아올 수 있습니다.
- [numpy \(<http://www.numpy.org/>\)](http://www.numpy.org/): numpy 패키지는 ndarray라는 파워풀한 자료구조를 지원하며, 각종 수치 컴퓨팅 관련 메소드를 지원합니다.
- [pandas \(<http://pandas.pydata.org/>\)](http://pandas.pydata.org/): 데이터 분석 라이브러리입니다.

## HTML과 CSS에 대한 기본 지식

- [W3School \(<http://www.w3schools.com/html/>\)](http://www.w3schools.com/html/)
- [codeacademy \(<https://www.codecademy.com/learn/web>\)](https://www.codecademy.com/learn/web)

### 1. 파일에서 index.html 파일을 만들어봅시다.

### 2. 바꿔봅시다!

1. 마우스 오른쪽 버튼 -> 검사
2. 마우스 오른쪽 버튼 -> Copy -> CSS Selector
3. body > div > div.desc-line.second

### 3. style 태그 대신 style.css 파일에 정리해둘 수도 있습니다.

<head>에 추가해주세요.

```
<link rel="stylesheet" href="style.css">
```

In [1]:

```
# 뷰티풀수프는 HTML에서 원하는 데이터를 쉽게 뽑아낼 수 있는 파이썬 라이브러리입니다.  
from bs4 import BeautifulSoup as bs
```

In [2]:

```
# 우리가 만든 index.html부터 살펴봅시다.  
soup = bs(open('./index.html'))
```

In [3]:

```
# soup 객체로부터 쉽게 parsing된 데이터들을 받아올 수 있습니다.
```

```
# title 태그를 받아옵니다.
```

```
print soup.title
```

```
# 태그 이름을 받아옵니다.
```

```
print soup.title.name
```

```
# 태그 내 문자열을 가져옵니다.
```

```
print soup.title.text
```

```
# 부모 태그를 가져옵니다.
```

```
print soup.title.parent.name
```

```
# p 태그를 가져옵니다.
```

```
print soup.p
```

```
<title>파이썬 클래스</title>
```

```
title
```

```
파이썬 클래스
```

```
head
```

```
<p>오늘은 <b id="crawling">크롤링</b>에 대해 배우고 있습니다.</p>
```

In [4]:

```
# 모든 a 태그 가져오기  
print soup.find_all('a')
```

```
[<a href="https://en.wikipedia.org/wiki/Web_crawler">위키피디아</a>,  
<a href="http://www.w3schools.com/html/">w3school 페이지에서 HTML, C  
SS에 대해 알아볼 수 있어요</a>]
```

In [5]:

```
# 클래스명으로 가져오기
print soup.select('.desc-box')

# id로 가져오기
print soup.select('#desc-important')

# CSS path로 가져오기
print soup.select('body > div > div.desc-line.first > span')

# 속성(attr)으로 가져오기
print soup.findAll('a', {'href': 'http://www.w3schools.com/html/'})

# 클래스도 속성 중 하나이기 때문에 이런식으로 가져올 수 있습니다.
print soup.findAll('div', {'class': 'desc-line'})
```

```
[<div class="desc-box">
<p>오늘은 <b id="crawling">크롤링</b>에 대해 배우고 있습니다.</p>
<div>

</div>
<span>크롤링에 대한 자세한 정의는 <a href="https://en.wikipedia.org/wik
i/Web_crawler">위키피디아</a>를 참고해보세요.</span>
<div class="desc-line first">
<span>웹 크롤링을 하려면 html과 css에 대해 먼저 알아야 합니다.</span>
<a href="http://www.w3schools.com/html/">w3school 페이지에서 HTML, C
SS에 대해 알아볼 수 있어요</a>
</div>
<div class="desc-line second">
<span><b id="desc-important">Javascript</b>는 웹페이지를 동적으로 만들어
주는 역할을 합니다.</span>
</div>
</div>]
[<b id="desc-important">Javascript</b>]
[<span>웹 크롤링을 하려면 html과 css에 대해 먼저 알아야 합니다.</span>]
[<a href="http://www.w3schools.com/html/">w3school 페이지에서 HTML,
CSS에 대해 알아볼 수 있어요</a>]
[<div class="desc-line first">
<span>웹 크롤링을 하려면 html과 css에 대해 먼저 알아야 합니다.</span>
<a href="http://www.w3schools.com/html/">w3school 페이지에서 HTML, C
SS에 대해 알아볼 수 있어요</a>
</div>, <div class="desc-line second">
<span><b id="desc-important">Javascript</b>는 웹페이지를 동적으로 만들어
주는 역할을 합니다.</span>
</div>]
```

In [6]:

```
# 자, 이제 필요한 HTML 태그를 뽑는 것 까지 가능합니다.  
# 태그 안에 속성(src, href 등)이나 text를 뽑아내는 방법에 대해 알아봅시다.  
  
# text만 뽑아내기  
second_line = soup.select('.desc-line.second')  
print second_line[0].text  
  
# a 태그 href 뽑아내기  
a_tag = soup.select('body > div > span > a')  
print a_tag[0].attrs  
print a_tag[0]['href']  
  
# img 태그 src 뽑아내기  
image_tag = soup.select('img')  
print image_tag[0].attrs  
print image_tag[0]['src']
```

Javascript는 웹페이지를 동적으로 만들어주는 역할을 합니다.

```
{'href': 'https://en.wikipedia.org/wiki/Web_crawler'}  
https://en.wikipedia.org/wiki/Web_crawler (https://en.wikipedia.o  
rg/wiki/Web_crawler)  
{'src': 'http://movement-as-medicine.com/wp-content/uploads/2014/  
01/baby-crawling.jpg'}  
http://movement-as-medicine.com/wp-content/uploads/2014/01/baby-c  
rawling.jpg (http://movement-as-medicine.com/wp-content/uploads/2  
014/01/baby-crawling.jpg)
```

이제 **beautifulsoup** 기본기는 닦은 것 같습니다. **requests** 라이브러리를 사용해 월드 와이드 웹 세상의 html을 받아옵시다.

In [7]:

```
# pip를 통해 설치한 requests를 import합니다.  
import requests
```

In [8]:

```
# 무료 이미지들을 크롤링해봅시다.  
res = requests.get('https://pixabay.com/en/')  
  
# dir()은 해당 객체의 사용 가능한 속성(attributes)를 list로 리턴해줍니다.  
# print dir(res)  
  
# HTTP 요청을 보내면 응답으로 Status Code가 오게 됩니다.  
# 200: 성공  
# 404: 페이지 없음  
# 500: 서버 오류  
  
# 200번이면 성공적으로 잘 받아온겁니다.  
print res.status_code
```

200

In [9]:

```
# bs(open('./index.html')) 했던 것 처럼,  
# index.html 자리에 res.text를 넣어줍니다.  
pixabay_html = bs(res.text)  
  
# 메인 페이지의 이미지를 받아서, 로컬 컴퓨터에 저장해보세요.  
# 이미지를 로컬 컴퓨터에 다운받으려면 urllib 라이브러리를 사용하면 됩니다.  
import urllib  
def download_image(img_src, filename):  
    res = urllib.urlretrieve(img_src, filename)  
    return res
```

## imdb에서 영화 정보 크롤링해보기

타겟 URL: <http://www.imdb.com/movies-coming-soon/2016-01/> (<http://www.imdb.com/movies-coming-soon/2016-01/>)

이곳에 들어가면 영화 제목, 장르, 평점, 감독, 배우 등의 정보가 있습니다. URL도 2015-01, 2015-02.. 이런식으로 바꿀 수 있습니다. 우선 2016-01 영화 정보를 함께 크롤링해봅시다.

In [14]:

```
# page를 순회하며 계속해서 사용할 코드이므로, 함수로 만들어봅시다.
def movie_crawler(url):
    res = requests.get(url)

    # status code를 확인하여, page가 없는 경우를 대비합니다.
    if res.status_code == 200:
        movie_html = bs(res.text)
    else:
        print 'error'
        return

    movies = movie_html.select('#main > div > div.list.detail > div')

    # title, image, running_time, score, genre, directors, actors
    # director와 actor는 여러명인 경우가 있는데
    # "/".join() 메소드를 사용하여, 하나로 묶어줍니다.
    table = []
    for movie in movies:
        row = []
        # ... 여기 코드를 입력해주세요.
        table.append(row)

    return table

# jan_2016 = movie_crawler('http://www.imdb.com/\
# movies-coming-soon/2016-01')
```

In [15]:

```
# 그러면 이제 2015-01부터 2015-12까지 영화정보를 긁어봅시다.
target_url = 'http://www.imdb.com/movies-coming-soon/{0}'
movie_total = []
for i in range(1,13):
    # string.zfill(2)을 사용해보세요. zero padding이 생깁니다.
    print "2015-" + str(i).zfill(2) + " crawling.."
    # movie_total += movie_crawler(target_url.\
    # format("2015-" + str(i).zfill(2)))
```

```
2015-01 crawling..
2015-02 crawling..
2015-03 crawling..
2015-04 crawling..
2015-05 crawling..
2015-06 crawling..
2015-07 crawling..
2015-08 crawling..
2015-09 crawling..
2015-10 crawling..
2015-11 crawling..
2015-12 crawling..
```

In [16]:

```
# 2016-01 데이터도 추가해줍시다.  
movie_total += jan_2016
```

In [17]:

```
# 데이터를 확인합니다.  
# print len(movie_total)  
# print movie_total[0]
```

In [18]:

```
# csv 파일로 바꿔봅시다.  
import csv  
  
# https://docs.python.org/2/library/csv.html#csv.reader  
# qoutechar, quoting 옵션이 뭔지 직접 해봅시다.  
with open('sample.csv', 'wb') as csvfile:  
    writer = csv.writer(csvfile, delimiter=',',  
                        quotechar='|', quoting=csv.QUOTE_MINIMAL)  
  
    writer.writerow(['Spam'] * 5 + ['Baked, Beans'])  
    writer.writerow(['Spam', 'Lovely, Spam', 'Wonderful Spam'])
```

In [19]:

```
# movie_total을 csv로 저장해봅시다.
```

In [20]:

```
# 저장된 csv 파일을 읽고, print 찍어봅시다.  
with open('sample.csv', 'rb') as csvfile:  
    movie_reader = csv.reader(csvfile, delimiter=',', quotechar='|')  
    for row in movie_reader:  
        print row
```

```
['Spam', 'Spam', 'Spam', 'Spam', 'Spam', 'Baked, Beans']  
['Spam', 'Lovely, Spam', 'Wonderful Spam']
```

In [21]:

```
# 자, 이제 파일 썬 데이터 분석을 pandas 소개합니다.  
from pandas import DataFrame
```

In [25]:

```
# movie_df = DataFrame(movie_total, columns=['title', 'image', \  
# 'running time', 'score', 'genre', 'directors', 'actors'])
```

In [26]:

```
# csv format으로 저장하기 (encoding utf-8)  
# movie_df.to_csv('./movie_from_df.csv', encoding='utf-8', index=False)
```

# 해보기

url: [http://www.imdb.com/search/name?birth\\_monthday=1-21&ref\\_=nm\\_ov\\_bth\\_monthday&refine=birth\\_monthday&start=51](http://www.imdb.com/search/name?birth_monthday=1-21&ref_=nm_ov_bth_monthday&refine=birth_monthday&start=51) ([http://www.imdb.com/search/name?birth\\_monthday=1-21&ref\\_=nm\\_ov\\_bth\\_monthday&refine=birth\\_monthday&start=51](http://www.imdb.com/search/name?birth_monthday=1-21&ref_=nm_ov_bth_monthday&refine=birth_monthday&start=51))

자신의 생일과 같은 사람들 영화관계자 200명을 DataFrame으로 읽고보고, csv 파일로도 저장해봅시다. (row => name, image, job, major\_work)

## iPython notebook 설치하기

numpy와 pandas는 파이썬 데이터 분석에 가장 많이 활용되는 패키지입니다. 앞으로 raw 데이터를 다루고 분석하는데 Pandas를 주로 사용하겠지만, Pandas는 numpy 자료구조에 dependency가 있습니다. numpy는 수치 연산에 강력한 ndarray 자료구조와 각종 메소드들을 지원합니다.

pip를 사용하여 numpy, pandas, ipython 패키지를 설치해봅시다.

```
pip install numpy
```

```
pip install pandas
```

```
pip install ipython[all]
```

참고: <https://ipython.org/ipython-doc/2/install/install.html> (<https://ipython.org/ipython-doc/2/install/install.html>)

## Pandas 맛보기

In [27]:

```
# 데이터를 가져와봅시다.  
import pandas
```

In [30]:

```
# http://wiki.stat.ucla.edu/socr/index.php/SOCR_Data_MLB_HeightsWeights  
# 위 사이트에 들어가서 표를 긁어서 복사(ctrl+c, 맥에서는 cmd+c) 합니다.  
baseball_players = pandas.read_clipboard()
```

In [31]:

```
# DataFrame의 head() 메소드를 활용해 잘 불었는지 확인해볼까요?  
baseball_players.head()
```

Out[31]:

	Name	Team	Position	Height(inches)	Weight(pounds)	Age
0	Adam_Donachie	BAL	Catcher	74	180	22.99
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78
3	Kevin_Millar	BAL	First_Baseman	72	210	35.43
4	Chris_Gomez	BAL	First_Baseman	73	188	35.71

In [32]:

```
# 긁어온 데이터를 csv로 저장해봅시다.  
baseball_players.to_csv('./baseball_player.csv', index=False)
```

In [33]:

```
# 저장한 .csv 데이터를 다시 읽어볼까요? 현재 폴더에 가서 잘 있는지 확인해봅시다.  
bp_data = pandas.read_csv('./baseball_player.csv')
```

In [34]:

```
bp_data.head()
```

Out[34]:

	Name	Team	Position	Height(inches)	Weight(pounds)	Age
0	Adam_Donachie	BAL	Catcher	74	180	22.99
1	Paul_Bako	BAL	Catcher	74	215	34.69
2	Ramon_Hernandez	BAL	Catcher	72	210	30.78
3	Kevin_Millar	BAL	First_Baseman	72	210	35.43
4	Chris_Gomez	BAL	First_Baseman	73	188	35.71

In [35]:

```
bp_data.columns
```

Out[35]:

```
Index(['Name', 'Team', 'Position', 'Height(inches)', 'Weight  
(pounds)',  
       'Age'],  
      dtype='object')
```

In [36]:

```
# Series  
heights = bp_data['Height(inches)']  
heights.head()
```

Out[36]:

```
0    74  
1    74  
2    72  
3    72  
4    73  
Name: Height(inches), dtype: int64
```

In [37]:

```
# numpy ndarray 자료구조로 value들을 리턴합니다.  
heights.values
```

Out[37]:

```
array([74, 74, 72, ..., 75, 75, 73])
```

In [38]:

```
# 이렇게 index도 받아올 수 있습니다.  
heights.index
```

Out[38]:

```
Int64Index([ 0, 1, 2, 3, 4, 5, 6, 7, 8,  
9,  
         ...  
1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032,  
1033],  
           dtype='int64', length=1034)
```

In [39]:

```
# 일단 Series 자료구조를 우리에게 익숙한 list로 바꿔봅시다.  
height_list = heights.tolist()
```

In [40]:

```
# 근데 농구선수 키가 왜 이렇게 작은걸까요? 인치를 센티미터로 바꿔봅시다.  
inch_to_cm = 2.54  
height_list[0]* inch_to_cm
```

Out[40]:

```
187.96000000000001
```

In [41]:

```
# height_list * 2.54 ?
[1,2,3]*2.54
```

```
-----
-----
TypeError                                 Traceback (most recent
call last)
<ipython-input-41-5afa9995de57> in <module>()
      1 # height_list * 2.54 ?
----> 2 [1,2,3]*2.54
```

```
TypeError: can't multiply sequence by non-int of type 'float'
```

In [42]:

```
# 예전에 배웠던 list comprehension을 사용할 수 있겠죠!
# 일단 여기까지하고, numpy, pandas, dataframe, series 차근차근 배워봅시다.
[i*inch_to_cm for i in height_list][0]
```

Out[42]:

```
187.96000000000001
```

In [43]:

```
# 보통 이렇게 numpy를 np로 줄여서 사용합니다.
# 사실 위에서 사용한 pandas도 편하게 pd로 줄여씁니다.
# numpy 패키지는 ndarray라는 파워풀한 자료구조를 지원하며,
# 각종 수치 컴퓨팅 관련 메소드를 지원합니다.
import numpy as np
```

In [44]:

```
arr = np.array([1,2,3,4])
print arr
print type(arr)
```

```
[1 2 3 4]
<type 'numpy.ndarray'>
```

In [45]:

```
np_heights = np.array(height_list)
np_heights
```

Out[45]:

```
array([74, 74, 72, ..., 75, 75, 73])
```

```
In [46]:
```

```
cm_heights = np_heights*inch_to_cm  
cm_heights
```

```
Out[46]:
```

```
array([ 187.96, 187.96, 182.88, ..., 190.5 , 190.5 , 185.4  
2])
```

## 연습문제 1.

야구선수들의 bmi(body mass index)를 구해봅시다.

$$bmi = \frac{weight(kg)}{height(m^2)}$$

```
In [47]:
```

```
# 1. 일단 bp_data로 부터 weight를 가져오고,  
# numpy ndarray로 만듭니다. pounds->kg으로 바꿔주세요.  
# => 정답:  
kg_weights = bp_data['Weight(pounds)'].values*0.453592
```

```
In [48]:
```

```
# ndarray자료구조인 weight, height를 받아 bmi를  
# 계산해서 리턴합니다. kg, m^2 단위 조심하세요!  
def cal_bmi(weight, height):  
    bmi = 0  
    # ...  
    # => 정답:  
    bmi = weight/height**2  
    return bmi
```

```
In [49]:
```

```
m_heights = cm_heights*0.01
```

```
In [50]:
```

```
bmi = cal_bmi(kg_weights, m_heights)
```

```
In [51]:
```

```
# 몸무게 적게 나가는 야구선수 구하기  
bmi[bmi<21]
```

```
Out[51]:
```

```
array([ 20.54255679, 20.54255679, 20.69282047, 20.69282047,  
       20.34343189, 20.34343189, 20.69282047, 20.15883472,  
       19.4984471 , 20.69282047, 20.9205219 ])
```

list와 numpy ndarray가 뚜렷하게 차이나는 점.

1. ndarray는 하나의 type만 넣을 수 있음.(type coercion)
2. 기본 연산(+,\*,-,+이 다른 의미를 가짐

In [52]:

```
print [1,2,3]+[4,5,6]
print [1,2,3]*3
```

  

```
[1, 2, 3, 4, 5, 6]
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

In [53]:

```
print np.array([1,2,3])+np.array([4,5,6])
print np.array([1,2,3])*3
```

  

```
[5 7 9]
[3 6 9]
```

## 2D Array (N-D Array)

In [54]:

```
# 일단 우리가 아는 list로 2D를 만들어봅시다.
baseball = [[180, 78.4],
            [215, 102.7],
            [210, 98.5],
            [188, 75.2]]
```

In [55]:

```
np_baseball = np.array(baseball)
np_baseball
```

Out[55]:

```
array([[ 180. ,    78.4],
       [ 215. ,   102.7],
       [ 210. ,    98.5],
       [ 188. ,    75.2]])
```

In [56]:

```
# shape는 몇행 몇열인지 알려줍니다.
np_baseball.shape
```

Out[56]:

```
(4, 2)
```

In [57]:

```
# 위의 데이터를 가지고 첫번째 column은 height,  
# 두번째 column은 weight인 2차 행렬을 만들어봅시다.  
baseball_player = np.array([kg_weights, m_heights])  
baseball_player.shape
```

In [58]:

```
(2, 1034)
```

In [59]:

```
print baseball_player
```

```
[[ 81.64656  97.52228  95.25432 ...,  92.98636  86.18248  88.4504  
4]  
[  1.8796     1.8796     1.8288   ...,   1.905      1.905      1.8542  
]]
```

In [60]:

```
# ndarray는 Transpose 메소드를 지원합니다.  
baseball_player = baseball_player.T
```

In [61]:

```
# 완성!  
baseball_player.shape
```

Out[61]:

```
(1034, 2)
```

In [62]:

```
# subset을 구할땐, 행->열 순입니다. 5번째 선수의 키는 이렇게..  
baseball_player[4][1]
```

Out[62]:

```
1.8542000000000003
```

In [63]:

```
# 키만 가지고 오고 싶을땐 이렇게..  
baseball_player[:,1]
```

Out[63]:

```
array([ 1.8796,  1.8796,  1.8288, ...,  1.905 ,  1.905 ,  1.854  
2])
```

In [64]:

```
# bmi까지 넣어 3차 행렬을 만들어봅시다.  
np_baseball_with_bmi = np.array([kg_weights, cm_heights, bmi])  
np_baseball_with_bmi = np_baseball_with_bmi.T
```

In [65]:

```
# numpy가 지원하는 착한 메소드들.. r  
np.mean(np_baseball_with_bmi[:,0])
```

Out[65]:

nan

In [66]:

```
# 숫자가 아닌 녀석들을 찾을땐 np.isnan  
np.isnan(np_baseball_with_bmi[:,0])
```

Out[66]:

```
array([False, False, False, ..., False, False, False], dtype=bool)
```

In [67]:

```
# 숫자인 녀석들을 찾아야하므로 앞에 ~  
~np.isnan(np_baseball_with_bmi[:,0])
```

Out[67]:

```
array([ True,  True,  True, ...,  True,  True,  True], dtype=bool)
```

In [68]:

```
no_nan_data = np_baseball_with_bmi\[  
~np.isnan(np_baseball_with_bmi[:,0])]
```

In [69]:

```
np.mean(no_nan_data[:,0])
```

Out[69]:

91.484632371732815

In [70]:

```
np.median(no_nan_data[:,0])
```

Out[70]:

90.718400000000003

In [71]:

```
np.std(no_nan_data[:,0])
```

Out[71]:

9.5169624670692663

In [72]:

```
np.corrcoef(no_nan_data[:,0], no_nan_data[:,1])
```

Out[72]:

```
array([[ 1.          ,  0.53188586],
       [ 0.53188586,  1.          ]])
```

In [73]:

```
no_nan_data
```

Out[73]:

```
array([[ 81.64656   ,  187.96      ,  23.11037639],
       [ 97.52228   ,  187.96      ,  27.60406069],
       [ 95.25432   ,  182.88      ,  28.48080465],
       ...,
       [ 92.98636   ,  190.5       ,  25.62295933],
       [ 86.18248   ,  190.5       ,  23.74810865],
       [ 88.45044   ,  185.42      ,  25.72686361]])
```

In [74]:

```
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

In [75]:

```
# histogram 그려기
# print plt.hist.__doc__
plt.hist(no_nan_data[:,1], color='indianred', alpha=0.5)
```

Out[75]:

```
(array([ 9.,  71.,  89., 318., 175., 263., 84., 14.,
7., 3.]),
array([ 170.18 , 174.244, 178.308, 182.372, 186.436, 190.5
,
194.564, 198.628, 202.692, 206.756, 210.82 ]),
<a list of 10 Patch objects>)
```

