example_basketball

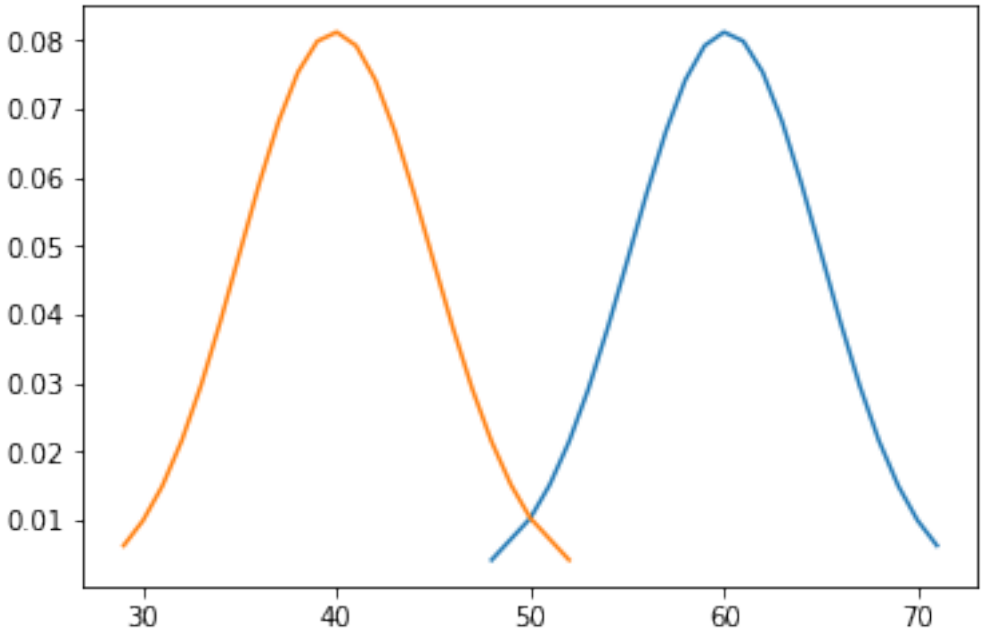
April 25, 2019

In [22]:
```
import matplotlib.pyplot as plt
from scipy.stats import binom
import numpy as np
import math

trials = 100
prob_two_point = 0.6
prob_three_point = 0.4

inputs_two_point = [binom.ppf(x, trials, prob_two_point) for x in np.arange(0, 1, 0.0
inputs_three_point = [binom.ppf(x, trials, prob_three_point) for x in np.arange(0, 1,
pdf_two_point = [binom.pmf(x, trials, prob_two_point) for x in inputs_two_point]
pdf_three_point = [binom.pmf(x, trials, prob_three_point) for x in inputs_three_point
plt.plot(inputs_two_point[1:], pdf_two_point[1:])
plt.plot(inputs_three_point[1:], pdf_three_point[1:])
```

Out[22]:
```
[<matplotlib.lines.Line2D at 0x1a17872a20>]
```

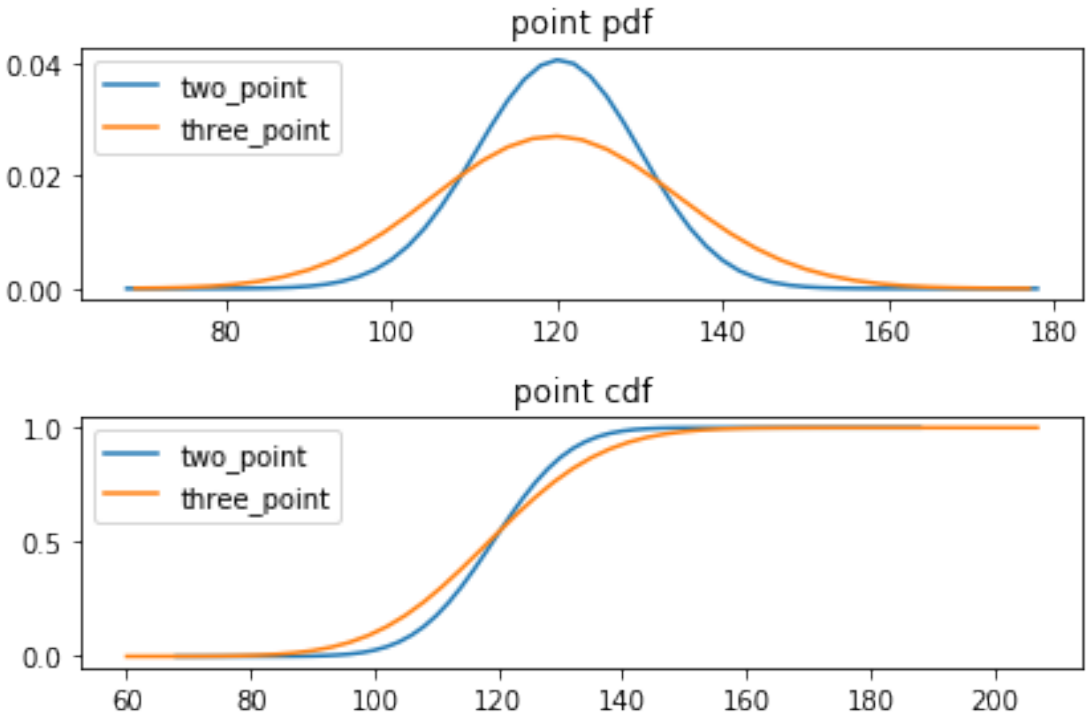


In [23]:
```
trials = 100
prob_two_point = 0.6
prob_three_point = 0.4

fig, axs = plt.subplots(2, 1)

shot_attempts = list(range(101))
points_twos = [2*s for s in shot_attempts]
points_threes = [3*s for s in shot_attempts]
pdf_two_point = [1 / 2 * binom.pmf(x, trials, prob_two_point) for x in shot_attempts]
pdf_three_point = [1 / 3 * binom.pmf(x, trials, prob_three_point) for x in shot_attem
axs[0].plot(points_twos[34:90], pdf_two_point[34:90], label='two_point')
axs[0].plot(points_threes[23:60], pdf_three_point[23:60], label='three_point')
axs[0].legend(loc='upper left')
axs[0].set_title('point pdf')

cdf_two_point = np.array(pdf_two_point).cumsum() * 2
cdf_three_point = np.array(pdf_three_point).cumsum() * 3
axs[1].plot(points_twos[34:95], cdf_two_point[34:95], label='two_point')
axs[1].plot(points_threes[20:70], cdf_three_point[20:70], label='three_point')
axs[1].legend(loc='upper left')
axs[1].set_title('point cdf')
plt.tight_layout()
```



In [24]:
```
def game_outcome_probabilities(shot_attempts, prob_points, prob_two, points=3):
    """probability of a three point only shooting team beating a two point only
    shooting team x_N_pt = number of baskets made for the N point shooting team
    Prob(three_point_team_wins) =
    Prob(two_point_team_points < three_point_team_points) =
    P(2*x_two_pt < 3*x_three_pt) =
    P(x_two_pt < 1.5*x_three_pt) =
    Sum[n=0..trials]P(x_two_pt < 1.5*x_three_pt|x_three_pt = n)P(x_three_pt = n)

    Args:
        shot_attempts (int): number of shots taken by each team
        prob_points (float): probability of score for the three point taking team
        prob_two (float): probability of score for the two point taking team
        points (float): number of points per basket for the 'prob_points' team

    Returns:
        dict: {'"points" wins': float, 'tie': float, 'two wins': float}
    """
    prob_points_wins = 0
    prob_tie = 0
    for made_points in np.arange(shot_attempts + 1):
        prob_made_points = binom.pmf(made_points, shot_attempts, prob_points)
        max_two_made_still_lose = math.floor(points / 2 * made_points)
        if 2 * max_two_made_still_lose == points * made_points:
            prob_tie = (prob_tie
                        + binom.pmf(max_two_made_still_lose, shot_attempts, prob_two)
                        * prob_made_points)
            max_two_made_still_lose = max_two_made_still_lose - 1
        if max_two_made_still_lose < 0:
            continue
        two_make_cdf = binom.cdf(max_two_made_still_lose, shot_attempts, prob_two)
        prob_points_wins = prob_points_wins + two_make_cdf * prob_made_points

    _points_wins = '{} wins'.format(points)
    res = {_points_wins: prob_points_wins,
           'tie': prob_tie,
           '2 wins': 1 - prob_points_wins - prob_tie}
    return res
```

In [21]:
```
shot_attempts = np.arange(0, 10000, 100)
points = 120
probs = [game_outcome_probabilities(shots, 0.01, 0.6, points) for shots in shot_attem
prob_differential = [prob['{} wins'.format(points)] - prob['2 wins'] for prob in probs
plt.plot(shot_attempts[1:], prob_differential[1:])
plt.title('Prob({} wins) - Prob(2 wins)'.format(points))
```

Out[21]: Text(0.5, 1.0, 'Prob(120 wins) - Prob(2 wins)')