Q1:

1. **min**: $Min(score)$ = 37.0
2. **max**: $Max(score)$ = 100.0

```
np.min(mid_score)
np.max(mid_score)
```

3. **first quartile Q1**: the 25^{th} percentile score = 68.0

   median: the 50^{th} percentile score = 77.0

   Third quartile Q3: the 75^{th} percentile score = 83.0

```
np.percentile(mid_score, 25)
np.percentile(mid_score, 50)
np.percentile(mid_score, 75)
```

4. **mean:**

   $mean(score) = \frac{1}{n}\sum_{i=1}^{n} x_i$ $(where\ n = the\ number\ of\ students,\ x_i = score\ of\ the\ i^{th}\ student$): = 76.715

```
round(np.mean(mid_score), 3)
```

5. **mode**: the score number that repeat most often = 77.0, 83.0

```
for score in mid_score:
	if count.has_key(score):
		count[score] += 1
	else:
		count[score] = 1
max = sorted(count.values())[len(count) - 1]
for score in count:
	if count[score] == max:
		print score
```

6. **empirical variance:** $S^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x^i - \bar{x})^2 = 173.279$

```
round(np.var(mid_score, ddof = 1), 3)
```

Q2:

1. **Compare the empirical variance before and after normalization.**

$$z = \frac{x - \mu}{\sigma}$$

(where x is raw score to be standardized, μ is mean of the population, σ is standard deviation)

The empirical variance before normalization is 173.279, after normalization is 1.0.

```
mid_score_z = preprocessing.scale(mid_score)
np.var(mid_score_z)
```

2. **Given original score of 90, what is the corresponding score after normalization?**

$$z = \frac{90 - \mu}{\sigma} = 1.009$$

```
round((90 - np.mean(mid_score)) / np.std(mid_score, ddof = 1), 3)
```

3. **Pearson's correlation coefficient between midterm scores and final scores is:**

$$cov(X, Y) = \frac{\sum_{n}^{i=1} (X_i - \bar{X})(Y_i - \bar{Y})}{n-1}$$

$$cor(X, Y) = \frac{cov(X,Y)}{\sigma_x \sigma_Y} = 0.544$$

```
round(np.corrcoef(mid_score, final_score)[1][0], 3)
```

4. **Covariance between midterm scores and final scores is:**

$$cov(X, Y) = \frac{\sum_{n}^{i=1} (X_i - \bar{X})(Y_i - \bar{Y})}{n-1} = 78.254$$

```
round(np.cov(mid_score, final_score)[1][0], 3)
```

Q3:

1. **the Jaccard coefficient of Citadel's Maester Library (CML) and Castle Black's library(CBL):**

$$sim_{Jaccard}(i,j) = \frac{q}{q+r+s} = \frac{58}{2+120+58} = 0.322$$

```
round (float(58) / float(2 + 120 + 58), 3)
```

2. **the minkowski distance of the two vectors with regard to different h values:**

$$d(i,j) = \sqrt[h]{|x_{i1} - x_{j1}|^h + |x_{i2} - x_{j2}|^h + \cdots + |x_{ip} - x_{jp}|^h}$$

(1) h = 1 (Manhattan distance) 6152

```
np.sum(np.fabs(CBL - CML))
```

(2) h = 2 (Euclidean) 715.328

```
round(np.sqrt(np.sum((CBL - CML)**2)), 3)
```

(3) h = 3 (Supremum = $\boldsymbol{max|x_{if} - x_{jf}|}$) 170

```
np.max(np.fabs(CBL - CML))
```

3. **the Cosine similarity between Citadel's Maester Library (CML) and Castle Black's:**

$$\cos(d_1, d_2) = \frac{d_1 * d_2}{|d_1||d_2|} = 0.841$$

```
round(np.sum(CML * CBL) / (np.linalg.norm(CBL) * np.linalg.norm(CML)), 3)
```

4. **the Kullback–Leibler divergence of these two libraries P(CML || CBL):**

$$D_{KL}(p(x)||q(x)) = \sum_{x \in X} p(x) \ln \frac{p(x)}{q(x)} = 0.201$$

```
round(np.sum((CML / np.sum(CML)) * np.log((CML / np.sum(CML)) / (CBL /
np.sum(CBL)))), 3)
```

Q4:

1. **the chi-square correlation value：**

sum = 150 + 40 + 15 + 3300 = 3505
bd = （150 + 40）*（150 + 15）/ 3505 = 8.94436519258
bnd = (150 + 40) * (40 + 3300) / 3505 = 181.055634807
nbd = (150 + 15) * (15 + 3300) / 3505 = 156.055634807
bndb = (15 + 3300) * (3300 + 40) / 3505 = 3158.94436519
(150 – 8.944) * (150 – 8.944) / 8.944 + (40 – 181.056) * (40 – 181.056) / 181.056 + (15 – 156.056) * (15 – 156.056)/ 156.056 + (3300 – 3158.944) * (3300 – 3158.944) / 3158.944 = 2468.183

```
bear_diaper = 150
bear_nodiaper = 40
nobear_diaper = 15
nobear_nodiaper = 3300
bear = bear_diaper + bear_nodiaper
diaper = bear_diaper + nobear_diaper
nobear = nobear_diaper + nobear_nodiaper
nodiaper = bear_nodiaper + nobear_nodiaper
sum = bear + nobear
b_d = float((bear * diaper)) / float(sum)
b_nd = float((bear * nodiaper)) / float(sum)
nb_d = float((nobear * diaper)) / float(sum)
nb_nd = float((nobear * nodiaper)) / float(sum)
a = np.square(bear_diaper - b_d) / b_d
b = np.square(bear_nodiaper - b_nd) / b_nd
c = np.square(nobear_diaper - nb_d) / nb_d
d = np.square(nobear_nodiaper - nb_nd) / nb_nd
chi = a + b + c + d
```