Q2:

```
n = 0
nsq = 466352
while(1): # if n*n > nsq break the while loop
    n += 1
    if(n*n > nsq):
        print (n)
        break
```

Answer:

```
============== RESTART: /Users/guoqch/Desktop/nc cw3/nc 2.py ==============
683
>>>
```

Q3:

```
def findn(): # define function
    n = 0
    nsq = 466352
    while(1): # if n*n > nsq break the while loop
        n += 1
        if(n*n > nsq):
            return n
            break

if __name__ == '__main__':
    print(findn())
```

Answer:

```
============== RESTART: /Users/guoqch/Desktop/nc cw3/nc 3.py ==============
683
>>>
```

Q4 :

```
import numpy as np
def getmatrics():
    n = int(input("Enter the n:")) # Input the integer
    matric = np.zeros((n,n)) # Create a empty matric
    for i in range(n):
        for j in range(n):
            matric[i][j] = 1.0/((i+1+1)*(j+1+1)) # Add element to the matric
    print (matric)

if __name__ == '__main__':
    getmatrics()
```

Answer:

```
=============== RESTART: /Users/guoqch/Desktop/nc cw3/nc 4.py ===============
Enter the n:6
[[0.25       0.16666667 0.125      0.1        0.08333333 0.07142857]
 [0.16666667 0.11111111 0.08333333 0.06666667 0.05555556 0.04761905]
 [0.125      0.08333333 0.0625     0.05       0.04166667 0.03571429]
 [0.1        0.06666667 0.05       0.04       0.03333333 0.02857143]
 [0.08333333 0.05555556 0.04166667 0.03333333 0.02777778 0.02380952]
 [0.07142857 0.04761905 0.03571429 0.02857143 0.02380952 0.02040816]]
>>>
```

Q5:
piSqiared.py

```
import math
def piSqiared(n):
    sum1 = 0.0
    array1 = [] #Empty array
    k = 1
    number = n
    PI = pow(math.pi,2) # Calculate the square of pi
    while n > 0: #while loop
        sum1 += (1/pow(n,2))
        n -= 1
    array1.append("%e"%number) # add the number to array
    array1.append(abs(PI - 6*sum1 )) # add the result to array
    print (array1)


for i in range(4): # Use for loop to get the anwsers
    piSqiared(pow(10, i+6))
```

Answer:

	Absolute error
10^6	5.999996737671154e-06
10^7	6.000058014876686e-07
10^8	5.40819087291311e-08
10^9	5.40819087291311e-08

When n=10^8 and n=10^9, it is obvious that the absolute error of them are equal.

Due to the reason is that 10^8 and 10^9 are too big. Therefore, the result of them are too small and it will exceed the range of the number , in other word, exceed the accuracy of the 16 digits. So number could not be displayed totally on the computer. Therefore, it seems that their result is equal.

Q5:

piSqiared_v2.py

```
import math
def piSqiared(n):
    sum1 = 0.0
    array1 = [] #Empty array
    k = 1
    number = n
    PI = pow(math.pi,2) # Calculate the square of pi
    while n > 0: #while loop
        sum1 += (1/pow(n,2))
        n -= 1
    array1.append("%e"%number) # add the number to array
    array1.append(abs(PI - 6*sum1 )) # add the result to array
    print (array1)


for i in range(4): # Use for loop to get the anwsers
    piSqiared(pow(10, i+6))
```

Answer :

```
>>>
========== RESTART: /Users/guoqch/Desktop/nc cw3/piSquared_v2.py ==========
['1.000000e+06', 5.999997000571966e-06]
['1.000000e+07', 5.999999697081648e-07]
['1.000000e+08', 5.999999963535174e-08]
['1.000000e+09', 6.000000496442226e-09]
```

	Absolute error
10^6	5.999997000571966e-06
10^7	5.999999697081648e-07
10^8	5.999999963535174e-08
10^9	6.000000496442226e-09

Compared with the original version, it is clear that the the absolute error in table 1 is smaller than table 2. Besides, the error of 10^8 and 10^9 is different.

The reason of that is if one number minus another number, it may have the remainder, it could cause the error. However, when the number adds another number, there is the carry bit. Therefore, there will not the error in calculation. Therefore, the original version has higher accuracy because the second version has subtraction.