```
import turtle

def main():
    randnum = readlist('randomnumbers copy.txt')
    #print(sorted(randnum))
    print('median:',get_median(randnum))
    print('mean:',get_mean(randnum))
    print('high:',get_high(randnum))
    print('low:',get_low(randnum))
    print('mode:', get_mode(randnum))
    print('percentile:', get_percentile(randnum,25))
    print('standard deviation:', standard_d(randnum))
    print('outliers:', outliers(randnum))
    graph(randnum,outliers(randnum),non_outliers(randnum,outliers(randnum)))

def write(graph, text, movement,pd=False):
            graph.up()
            graph.forward(movement)
            graph.write(text/4+50,False, align="center")
            graph.forward(-movement)
            if pd == True:
                    graph.down()

def graph(randnum,outliers,non_outliers):
        wn = turtle.Screen()
        graph = turtle.Turtle()
        graph.ht()
        graph.speed(0)
        graph.left(90)
        Q3 = 4*(get_percentile(randnum,75)-50)
        Q1 = 4*(get_percentile(randnum,25)-50)
        median= 4*(get_median(randnum)-50)
        low = get_low(non_outliers)
        high = get_high(non_outliers)
        height = 17
        graph.up()
        for i in outliers:
                graph.goto(4*(i-50),0)
                graph.dot(3, "black")
                graph.goto(4*(i-50),height+5)
                graph.write(i,False, align="left")
        graph.fillcolor('red')
        graph.goto(low,-height)
        graph.down()
        graph.goto(low,height)
        write(graph, low, 5)
        graph.goto(low,0)
        graph.down()
        graph.goto(Q1,0)
        graph.up()
        graph.goto(median,-height)
        graph.begin_fill()
        graph.down()
        graph.goto(Q1,-height)
        graph.goto(Q1,height)
        write(graph, Q1, 5,True)
        graph.goto(median,height)
        graph.end_fill()
        write(graph, median, 5, True)
        graph.fillcolor('green')
        graph.goto(median,height)
        graph.begin_fill()
        graph.goto(Q3,height)
        write(graph, Q3, 5,True)
        graph.goto(Q3,-height)
        graph.goto(median,-height)
        graph.goto(median,height)
        graph.end_fill()
        graph.up()
        graph.goto(Q3,0)
        graph.down()
        graph.goto(high,0)
        graph.goto(high,-height)
        graph.goto(high,height)
        write(graph, high, 5)
        for i in range(0,416,16):
                graph.goto(-200+i,-45)
                graph.down()
                graph.goto(-200+i,-35)
                graph.up()
                graph.goto(-200+i,-60)
                graph.write(int(i/4), False, align="center")
        graph.goto(200,-45)
        graph.down()
        graph.goto(-200,-45)


def outliers(randnum):
        outliers = []
        Q3 = get_percentile(randnum,75)
        Q1 = get_percentile(randnum,25)
        IQR = Q3 - Q1
        for num in randnum:
                if num > (1.5*IQR+Q3):
                        outliers.append(num)
                        print(num)
                elif num < (Q1 - 1.5 *IQR):
                        outliers.append(num)
        return outliers

def non_outliers(randnum,outliers):
        non_outliers = []
        for num in randnum:
                if num in outliers:
                        pass
                else:
                        non_outliers.append(4*(num-50))
        return non_outliers

def readlist(file):
    read = open(file,'r')
    numbers = []
    for line in read:
        numbers.append(float(line.strip()))
    return numbers

def get_mode(randnum):
    numbers = {}
    for i in randnum:
        try:
            numbers[i] += 1
        except(KeyError):
            numbers[i] = 1
    nums = numbers.keys()
    max = 0
    for num in nums:
        if numbers[num] > max:
            max = numbers[num]
    for num in nums:
        if numbers[num] == max:
            return num

def get_high(randnum):
    ahigh = randnum[0]
    for num in randnum:
        if num > ahigh:
            ahigh = num
    return ahigh

def get_low(randnum):
    alow = randnum[0]
    for num in randnum:
        if num < alow:
            alow = num
    return alow

def get_percentile(rand,percent):
    randnum = rand[:]
    for a in range(int(len(randnum)/100.*(100-percent)-1)):
        randnum.remove(get_high(randnum))
    avg = 0
    if len(rand)%2 == False:
        avg += randnum.pop(randnum.index(get_high(randnum)))
        avg += randnum.pop(randnum.index(get_high(randnum)))
        avg = avg/2.
        return avg
    return get_high(randnum)

def get_mean(randnum):
    mean = 0
    for num in randnum:
        mean += num
    return mean/float(len(randnum))

def get_median(rand):
    return get_percentile(rand,50)
    avg = 0
    randnum = rand[:]
    for a in range(int(len(randnum)/2-1)):
        randnum.remove(get_high(randnum))
        randnum.remove(get_low(randnum))
    if len(randnum)%2 == True:
        randnum.remove(get_high(randnum))
        randnum.remove(get_low(randnum))
        for i in randnum:
            avg += i
    else:
        for i in randnum:
            avg += i
        avg = avg/2.
    return avg

def standard_d(numlist):
    mean = get_mean(numlist)
    sigma2 = 0
    for x in numlist:
        sigma2 += (x-mean)**2.
    sigma2 = sigma2/len(numlist)
    sigma = sigma2**(1./2.)
    return sigma
```

```python
if __name__ == "__main__":
    main()
```