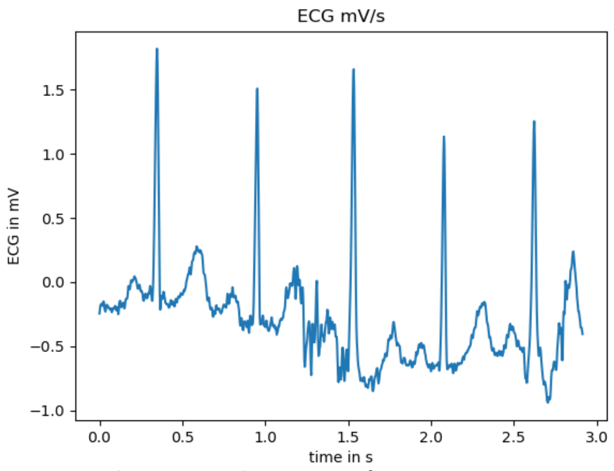
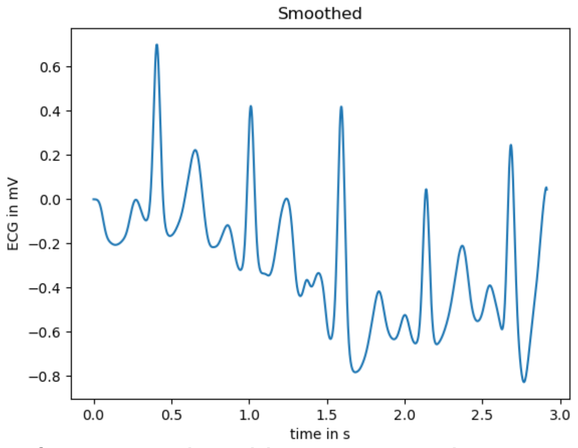
01- Basic ECG Analysis

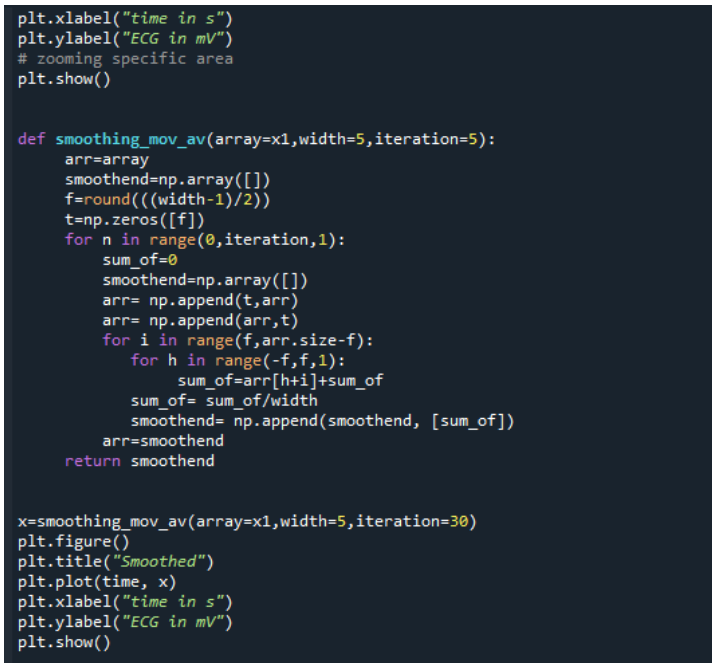
Create a PQRST block detector for the smoothed ECG signal.

1- Create a moving smoothing function. Please smooth the signal to an extent where it does not lose its characteristics.
2- Determine PQRST keypoint on the smoothed signal.
3- Calculate the time intervals of the following intervals
In the same block: PQ, QR, RS, ST, PT
In sequential blocks: PP, QQ, TP
4- Submit python script about above instructions
5- Submit a report containing images showing that the specified operations have been carried out. Your report must be in PDF format.

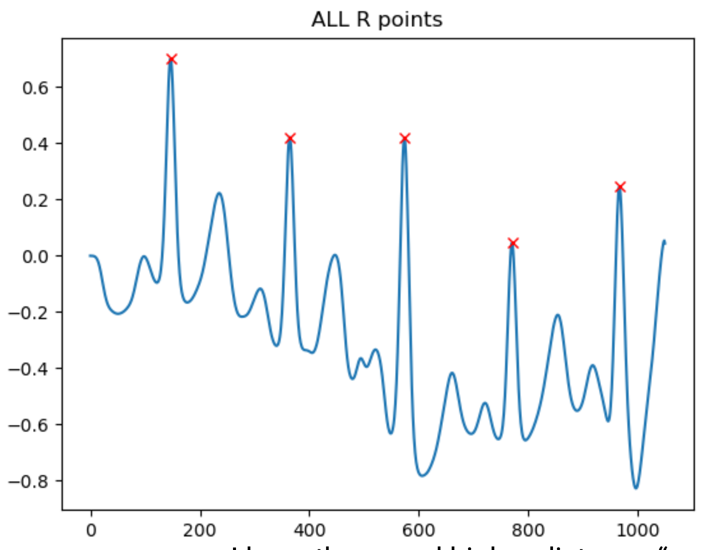


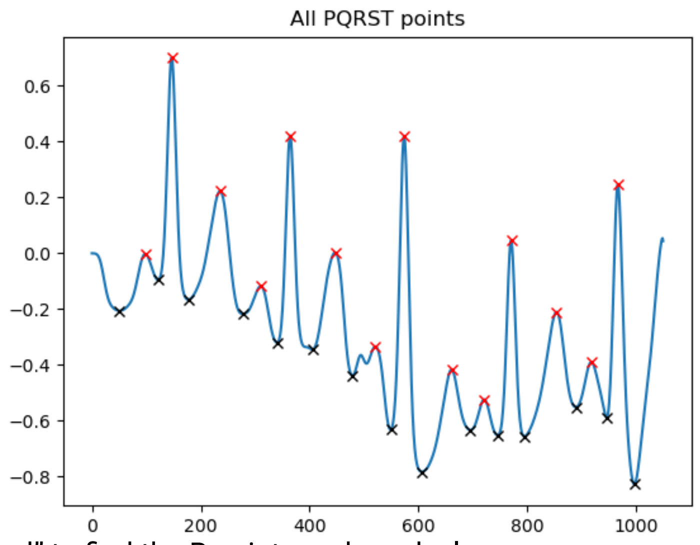


I have used a general use moving average function with width, iteration and not shortening with added zeros as buffer. I have used width of 5 to make sure the signal is not smoothing over the signal. Then I used the iteration to smoothen the signal for PQRST detection.

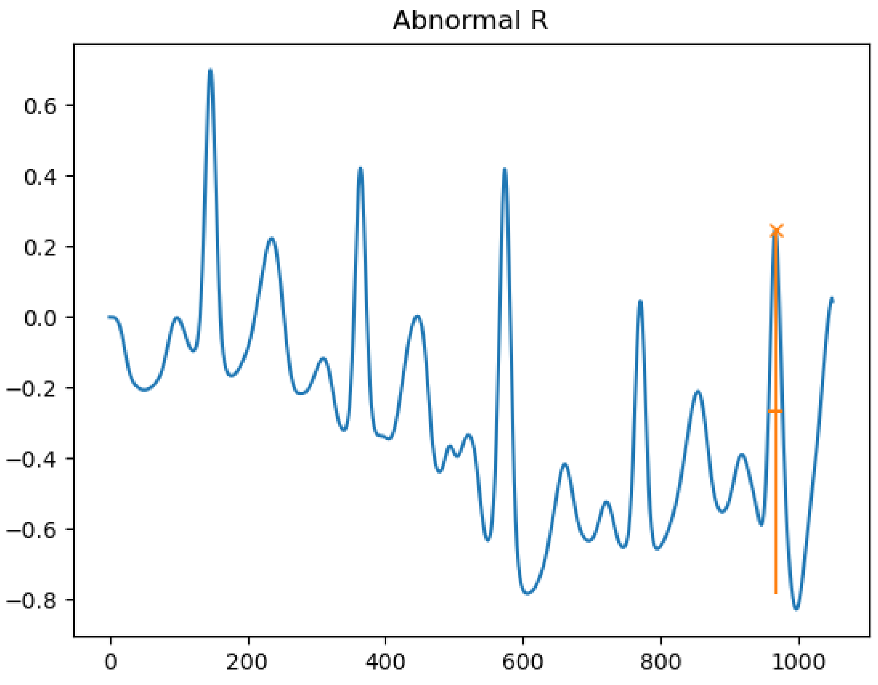


```
plt.xlabel("time in s")
plt.ylabel("ECG in mV")
# zooming specific area
plt.show()


def smoothing_mov_av(array=x1,width=5,iteration=5):
     arr=array
     smoothend=np.array([])
     f=round(((width-1)/2))
     t=np.zeros([f])
     for n in range(0,iteration,1):
         sum_of=0
         smoothend=np.array([])
         arr= np.append(t,arr)
         arr= np.append(arr,t)
         for i in range(f,arr.size-f):
            for h in range(-f,f,1):
                 sum_of=arr[h+i]+sum_of
            sum_of= sum_of/width
            smoothend= np.append(smoothend, [sum_of])
         arr=smoothend
     return smoothend


x=smoothing_mov_av(array=x1,width=5,iteration=30)
plt.figure()
plt.title("Smoothed")
plt.plot(time, x)
plt.xlabel("time in s")
plt.ylabel("ECG in mV")
plt.show()
```

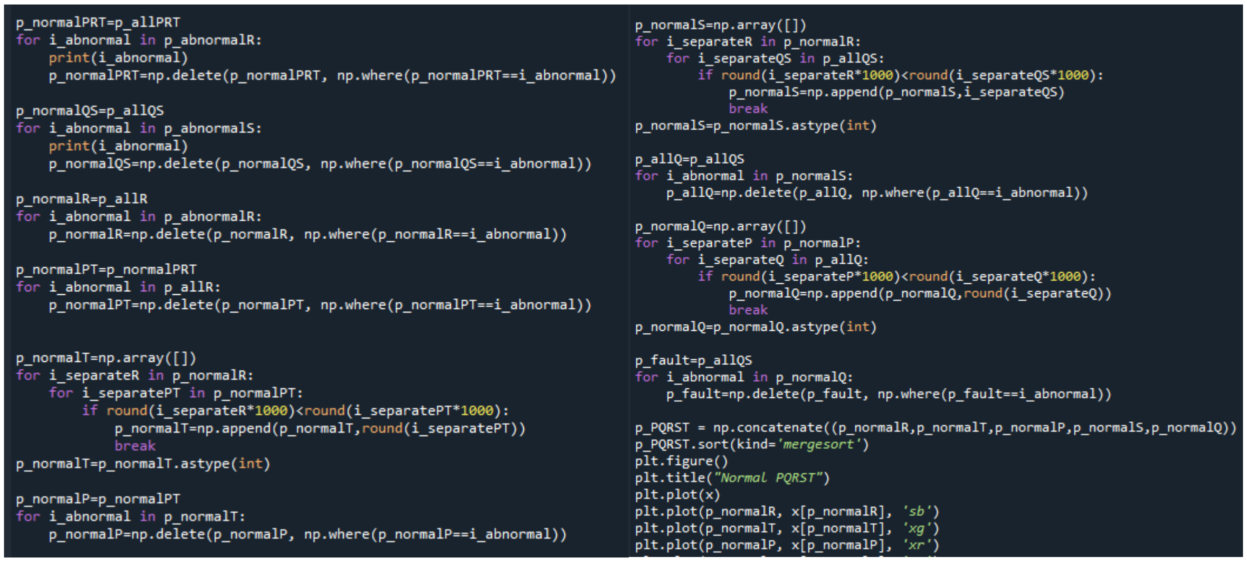




I have then used higher distance “peak_find” to find the R points and used a lesser distance “peak_find” to find P, R , T waves then used “peak_find” function to find Q and S waves.



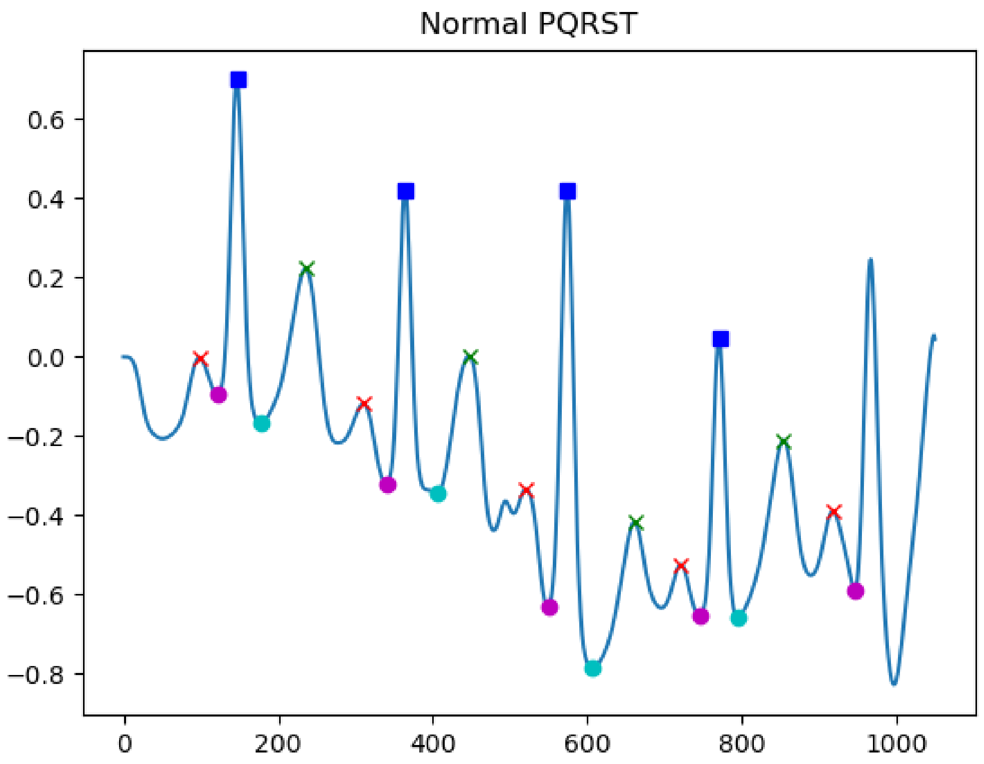
I have used peak_find to abnormal R peaks and Abnormal S peaks which I have eliminated them for accurate readings.

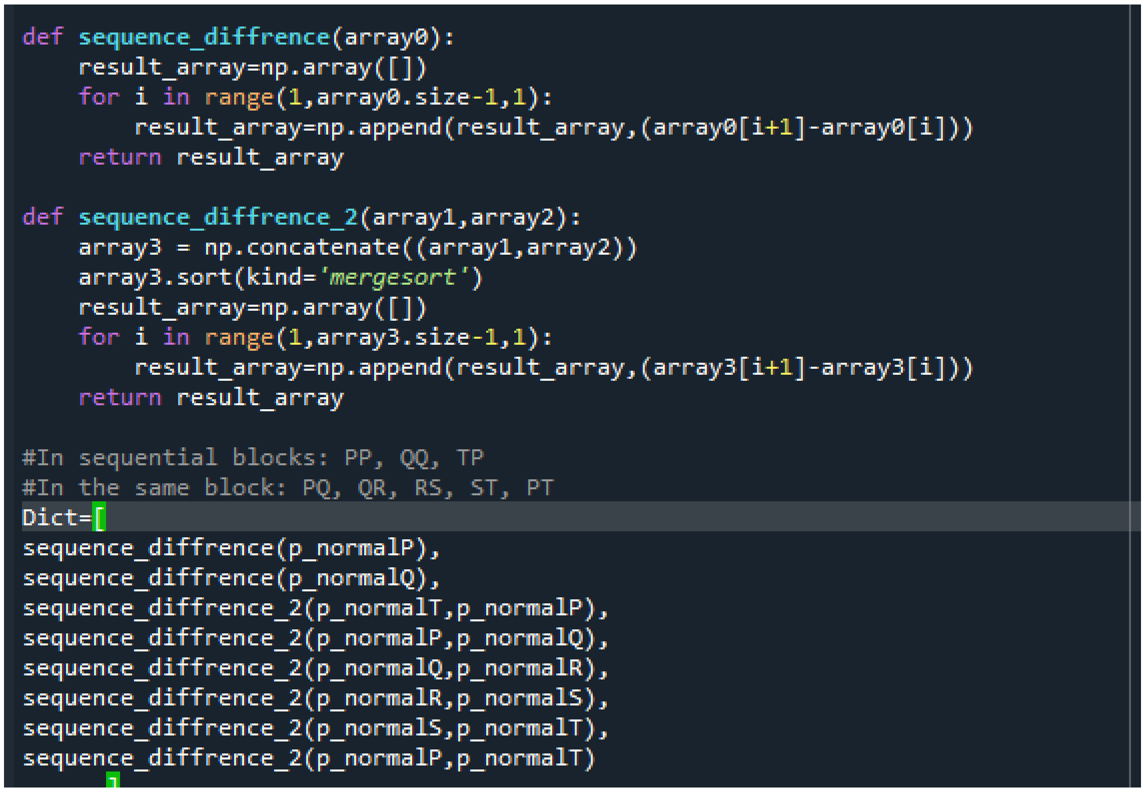


```python
p_normalPRT=p_allPRT
for i_abnormal in p_abnormalR:
    print(i_abnormal)
    p_normalPRT=np.delete(p_normalPRT, np.where(p_normalPRT==i_abnormal))

p_normalQS=p_allQS
for i_abnormal in p_abnormalS:
    print(i_abnormal)
    p_normalQS=np.delete(p_normalQS, np.where(p_normalQS==i_abnormal))

p_normalR=p_allR
for i_abnormal in p_abnormalR:
    p_normalR=np.delete(p_normalR, np.where(p_normalR==i_abnormal))

p_normalPT=p_normalPRT
for i_abnormal in p_allR:
    p_normalPT=np.delete(p_normalPT, np.where(p_normalPT==i_abnormal))


p_normalT=np.array([])
for i_separateR in p_normalR:
    for i_separatePT in p_normalPT:
        if round(i_separateR*1000)<round(i_separatePT*1000):
            p_normalT=np.append(p_normalT,round(i_separatePT))
            break
p_normalT=p_normalT.astype(int)

p_normalP=p_normalPT
for i_abnormal in p_normalT:
    p_normalP=np.delete(p_normalP, np.where(p_normalP==i_abnormal))
```

```python
p_normalS=np.array([])
for i_separateR in p_normalR:
    for i_separateQS in p_allQS:
        if round(i_separateR*1000)<round(i_separateQS*1000):
            p_normalS=np.append(p_normalS,i_separateQS)
            break
p_normalS=p_normalS.astype(int)

p_allQ=p_allQS
for i_abnormal in p_normalS:
    p_allQ=np.delete(p_allQ, np.where(p_allQ==i_abnormal))

p_normalQ=np.array([])
for i_separateP in p_normalP:
    for i_separateQ in p_allQ:
        if round(i_separateP*1000)<round(i_separateQ*1000):
            p_normalQ=np.append(p_normalQ,round(i_separateQ))
            break
p_normalQ=p_normalQ.astype(int)

p_fault=p_allQS
for i_abnormal in p_normalQ:
    p_fault=np.delete(p_fault, np.where(p_fault==i_abnormal))

p_PQRST = np.concatenate((p_normalR,p_normalT,p_normalP,p_normalS,p_normalQ))
p_PQRST.sort(kind='mergesort')
plt.figure()
plt.title("Normal PQRST")
plt.plot(x)
plt.plot(p_normalR, x[p_normalR], 'sb')
plt.plot(p_normalT, x[p_normalT], 'xg')
plt.plot(p_normalP, x[p_normalP], 'xr')
```

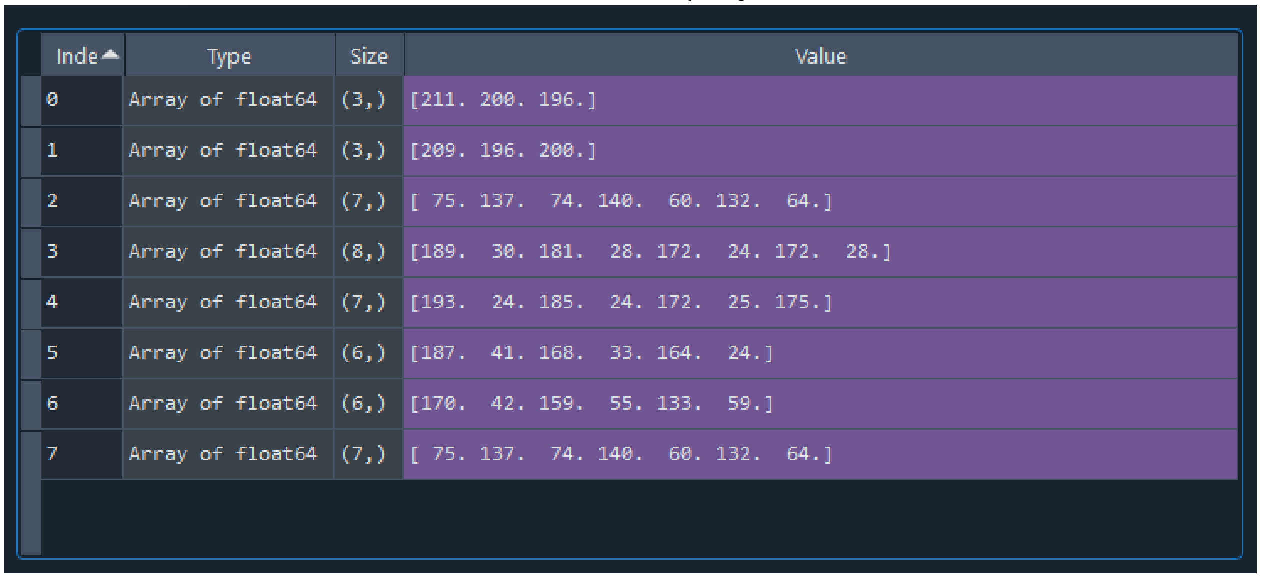
First separating PRT and abnormal R to test then splitting them all each elements by using R as a guide. I have then used merge sort to show each point highlighted right below.





```python
def sequence_diffrence(array0):
    result_array=np.array([])
    for i in range(1,array0.size-1,1):
        result_array=np.append(result_array,(array0[i+1]-array0[i]))
    return result_array

def sequence_diffrence_2(array1,array2):
    array3 = np.concatenate((array1,array2))
    array3.sort(kind='mergesort')
    result_array=np.array([])
    for i in range(1,array3.size-1,1):
        result_array=np.append(result_array,(array3[i+1]-array3[i]))
    return result_array

#In sequential blocks: PP, QQ, TP
#In the same block: PQ, QR, RS, ST, PT
Dict=[
sequence_diffrence(p_normalP),
sequence_diffrence(p_normalQ),
sequence_diffrence_2(p_normalT,p_normalP),
sequence_diffrence_2(p_normalP,p_normalQ),
sequence_diffrence_2(p_normalQ,p_normalR),
sequence_diffrence_2(p_normalR,p_normalS),
sequence_diffrence_2(p_normalS,p_normalT),
sequence_diffrence_2(p_normalP,p_normalT)
     ]
```

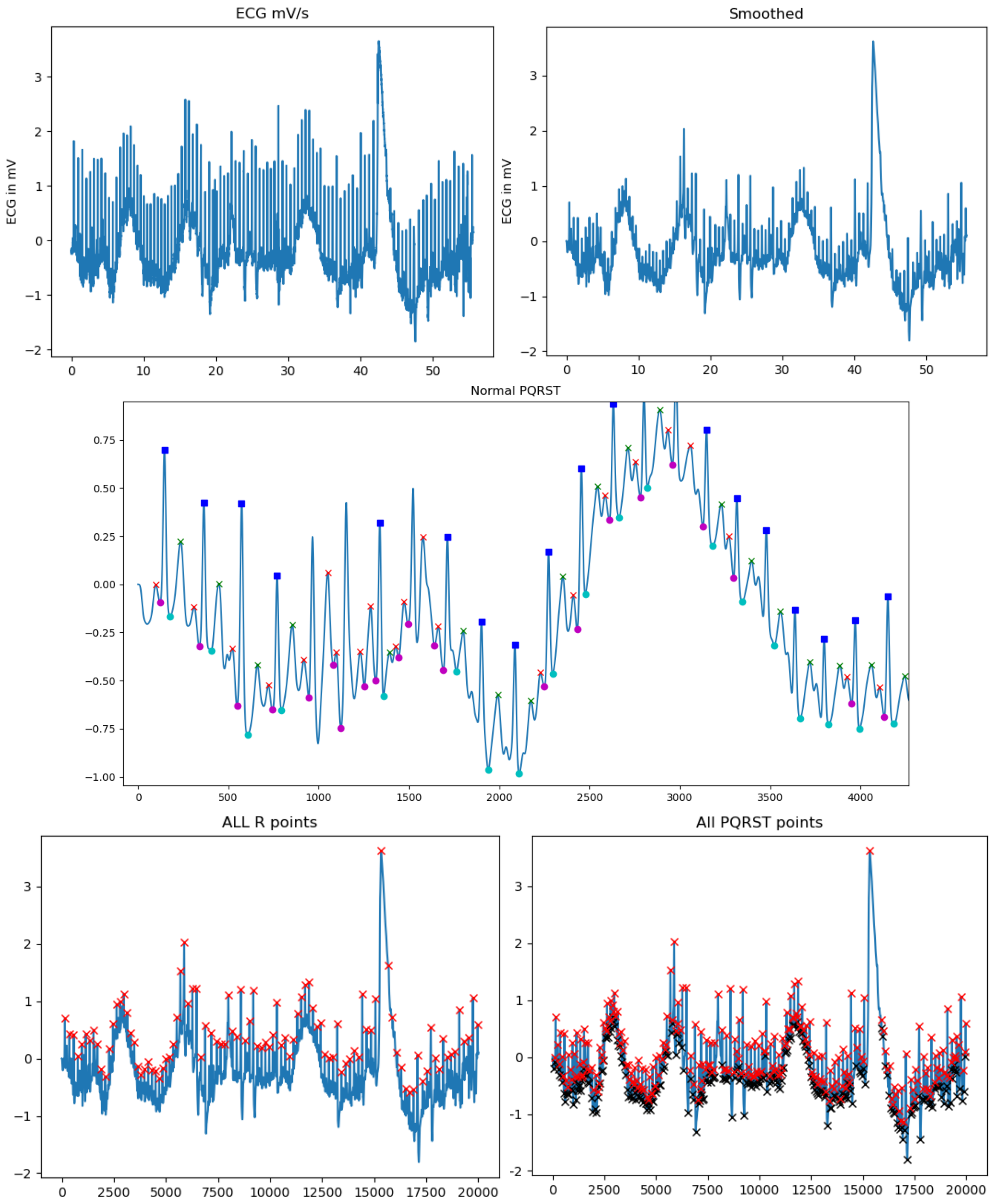
Writing two functions to handle each requested time intervals, I used concatenate to make an array and take differences of time between them. Using Spyder development tool as I have shown below, taken time intervals are in 360Hz, Sampling size of 0 to are as follows.

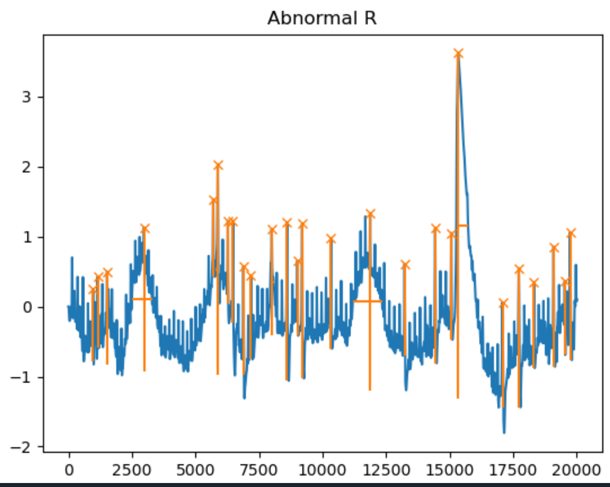


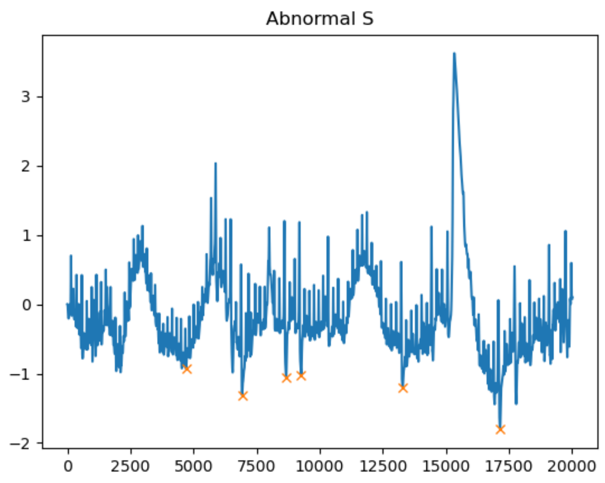
Inde	Type	Size	Value
0	Array of float64	(3,)	[211. 200. 196.]
1	Array of float64	(3,)	[209. 196. 200.]
2	Array of float64	(7,)	[75. 137. 74. 140. 60. 132. 64.]
3	Array of float64	(8,)	[189. 30. 181. 28. 172. 24. 172. 28.]
4	Array of float64	(7,)	[193. 24. 185. 24. 172. 25. 175.]
5	Array of float64	(6,)	[187. 41. 168. 33. 164. 24.]
6	Array of float64	(6,)	[170. 42. 159. 55. 133. 59.]
7	Array of float64	(7,)	[75. 137. 74. 140. 60. 132. 64.]

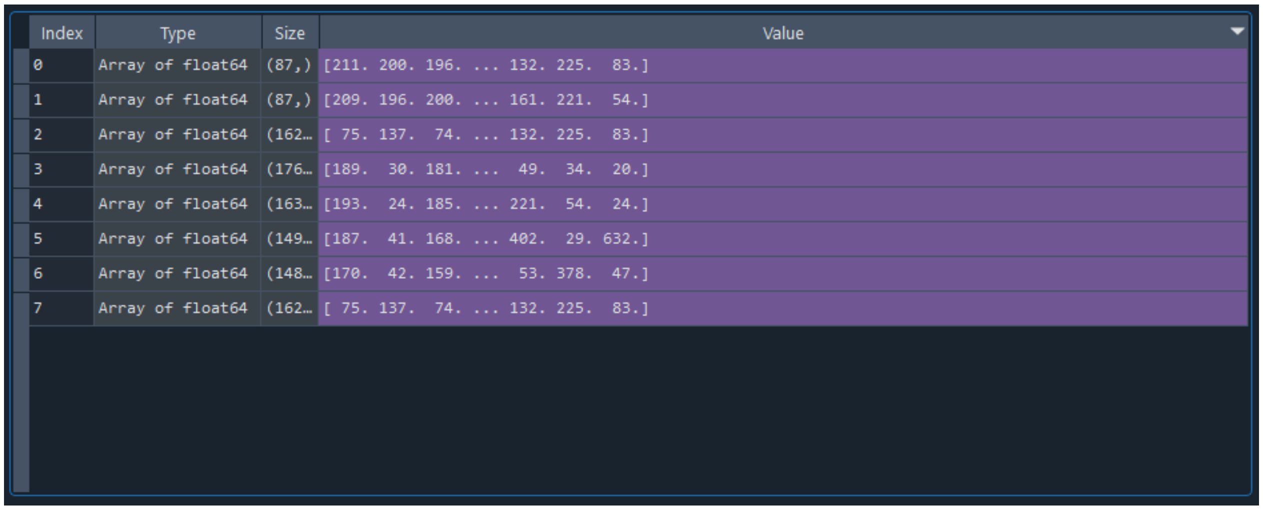
P to P ,Q to Q ,T to P ,P to Q ,Q to R ,R to S ,S to T

Right Below are in 360Hz, Sampling size of 0 to 20000


ECG mV/s
ECG in mV
Smoothed
ECG in mV
Normal PQRST
ALL R points
All PQRST points







Index	Type	Size	Value
0	Array of float64	(87,)	[211. 200. 196. ... 132. 225. 83.]
1	Array of float64	(87,)	[209. 196. 200. ... 161. 221. 54.]
2	Array of float64	(162…	[75. 137. 74. ... 132. 225. 83.]
3	Array of float64	(176…	[189. 30. 181. ... 49. 34. 20.]
4	Array of float64	(163…	[193. 24. 185. ... 221. 54. 24.]
5	Array of float64	(149…	[187. 41. 168. ... 402. 29. 632.]
6	Array of float64	(148…	[170. 42. 159. ... 53. 378. 47.]
7	Array of float64	(162…	[75. 137. 74. ... 132. 225. 83.]