

```

1  #versione 1.0 multithread
2  #nome del file : pagellaServerMulti_V01.py
3  import socket
4  from threading import Thread
5  import json
6  import pprint
7
8  SERVER_ADDRESS = '127.0.0.1'
9  SERVER_PORT = 22001
10
11
12 def ricevi_comandi(sock_service):
13     print(["avviato"])
14     while True:
15         data=sock_service.recv(1024)
16         if not data:
17             break
18         data=data.decode()
19         data=json.loads(data)
20         # completare:
21         #1. recuperare studente, materia, voto e assenze
22         studente=data['studente']
23         materia=data['materia']
24         voto=int(data['voto'])
25         assenze=int(data['assenze'])
26
27         #2. restituire studente, materia e una valutazione testuale sapendo che :
28         # voto < 4 Gravemente insufficiente
29         # voto <= 5 Insufficiente
30         # voto = 6 Sufficiente
31         # voto = 7 Discreto
32         # voto compreso tra 8 e 9 Buono
33         # voto = 10 Ottimo
34         if voto<4:
35             valutazione="Gravemente insufficiente"
36         elif voto<=5:
37             valutazione="Insufficiente"
38         elif voto==6:
39             valutazione="Sufficiente"
40         elif voto==7:
41             valutazione="Discreto"
42         elif voto>=8 and voto<=9:
43             valutazione="Buono"
44         elif voto==10:
45             valutazione="Ottimo"
46         messaggio={'studente':studente,
47                   'materia':materia,
48                   'valutazione':valutazione}
49         print("Dati inviati al client:")
50         print(messaggio)
51         messaggio=json.dumps(messaggio)
52         sock_service.sendall(messaggio.encode("UTF-8"))
53     sock_service.close()

```

```

55 | def ricevi_comandi2(sock_service):
56 |     print("avviato")
57 |     while True:
58 |         data=sock_service.recv(1024)
59 |         if not data:
60 |             break
61 |         data=data.decode()
62 |         data=json.loads(data)
63 |         # completare:
64 |         #1. recuperare studente e pagella
65 |         studente=data['studente']
66 |         pagella=data['pagella']
67 |         #2. restituire studente, media dei voti e somma delle assenze :
68 |         assenze=0
69 |         media=0
70 |         for i,p in enumerate(pagella):
71 |             media+=int(p[1])
72 |             assenze+=int(p[2])
73 |         media=media/i
74 |         messaggio={'studente':studente,
75 |                   'media':media,
76 |                   'assenze':assenze}
77 |         print("Dati inviati al client:")
78 |         print(messaggio)
79 |         messaggio=json.dumps(messaggio)
80 |         sock_service.sendall(messaggio.encode("UTF-8"))
81 |     sock_service.close()

```

```

83 | def ricevi_comandi3(sock_service):
84 |     print("avviato")
85 |     while True:
86 |         data=sock_service.recv(1024)
87 |         if not data:
88 |             break
89 |         data=data.decode()
90 |         data=json.loads(data)
91 |         # completare:
92 |         #1. recuperare il tabellone
93 |         pp=pprint.PrettyPrinter(indent=4)
94 |         #2. restituire una lista di dizionari : studente, media dei voti e somma delle assenze
95 |         tabellone=[]
96 |         for stud in data:
97 |             pagella=data[stud]
98 |             assenze=0
99 |             media=0
100 |             for i,p in enumerate(pagella):
101 |                 media+=int(p[1])
102 |                 assenze+=int(p[2])
103 |             media=media/i
104 |             messaggio={'studente':stud,
105 |                       'media':media,
106 |                       'assenze':assenze}
107 |             tabellone.append(messaggio)
108 |         print("Dati inviati al client:")
109 |         pp.pprint(tabellone)
110 |         messaggio=tabellone
111 |         messaggio=json.dumps(messaggio)
112 |         sock_service.sendall(messaggio.encode("UTF-8"))
113 |     sock_service.close()

```

```

115 def ricevi_conessioni(sock_listen):
116     while True:
117         sock_service, addr_client = sock_listen.accept()
118         print("\nConnessione ricevuta da " + str(addr_client))
119         print("\nCreo un thread per servire le richieste ")
120         try:
121             Thread(target=ricevi_comandi1,args=(sock_service,addr_client)).start()
122             #Thread(target=ricevi_comandi2,args=(sock_service,addr_client)).start()
123             #Thread(target=ricevi_comandi3,args=(sock_service,addr_client)).start()
124         except:
125             print("il thread non si avvia")
126             sock_listen.close()
127
128 def avvia_server(SERVER_ADDRESS,SERVER_PORT):
129     sock_listen=socket.socket()
130     sock_listen.setsockopt(socket.SOL_SOCKET,socket.SO_REUSEADDR,1)
131     sock_listen.bind((SERVER_ADDRESS,SERVER_PORT))
132     sock_listen.listen(5)
133     print("Server in ascolto su %s." %str((SERVER_ADDRESS,SERVER_PORT)))
134     ricevi_conessioni(sock_listen)
135
136 if __name__=='__main__':
137     avvia_server([SERVER_ADDRESS,SERVER_PORT])

```