```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import sys

# Caricamento csv della simulazione Pyrosim con nominazione finale "nomefile_devc.csv"
df = pd.read_csv('CHLArena_020523_devc.csv')
# Selezione della prima riga del DataFrame che conterrà i nomi delle colonne
col_names = df.iloc[0]
# Si rinominano le colonne del DataFrame
df = df.rename(columns=dict(zip(df.columns, col_names)))
# Selezione di tutte le righe, tranne la prima riga di intestazione
df = df.iloc[1:]
# Si fa un cast dei dati del DataFrame a float, poichè sono di default di tipo str
df = df.astype(float)


# Le soglie per i modelli di calcolo sono fissate dalla Tabella M.3-2 del Codice di Prevenzione Incendi


# ASET temperatura piano secondo
soglia = 60
# Creazione di una lista dei sensori di temperatura, in cui "^" serve per indicare alla
# regex che deve selezionare tutto quello che inizia con l'argomento specificato
sensori = df.filter(regex='^T_P2').columns.tolist()
# Selezione dei valori corrispondenti alle colonne desiderate
df_2 = df.loc[:, sensori]
# Creazione di un terzo DataFrame in cui fare la ricerca dei valori maggiori della soglia
# In df_3 è presente il valore se è superiore alla soglia oppure NaN
df_3 = df_2[df_2 >= soglia]
# Conteggio dei valori superiori alla soglia
conteggio = df_3.count().sum()
if conteggio > 0:
    # Per cercare l'indice minimo si setta al massimo intero rappresentabile una variabile index_min
    index_min = sys.maxsize
    for colonna in df_3.columns:
        index = df_3[colonna].first_valid_index()
        if index != None:
            if index < index_min:
                index_min = index
    print("L'ASET risultato dall'analisi del modello del calore, analizzando gli output della temperatura massima di esposizione al piano secondo, è:{:.2f}".format(df['Time'][index_min] )+ "secondi")
# Se conteggio è zero allora vuol dire che non c'è nessun valore che ha superato la soglia
# Pertanto, nel tempo di simulazione non si sono verificate condizioni incapacitanti
else:
    print("I valori risultanti dall'analisi del modello del calore, che analizza le temperatura massima di esposizione al piano secondo, sono ininfluenti")




# ASET temperatura piano primo e parterre
soglia = 60
sensori = df.filter(regex='^T(_P1)' and '^T(_PT)').columns.tolist()
df_2 = df.loc[:, sensori]
df_3 = df_2[df_2 >= soglia]
conteggio = df_3.count().sum()
if conteggio > 0:
    index_min = sys.maxsize
    for colonna in df_3.columns:
        index = df_3[colonna].first_valid_index()
        if index != None:
            if index < index_min:
                index_min = index
    print("L'ASET risultato dall'analisi del modello del calore, analizzando gli output della temperatura massima di esposizione al piano parterre e primo,
```

```
        è:{:.2f}".format(df['Time'][index_min] )+ "secondi")
64  else:
65      print("I valori risultanti dall'analisi del modello del calore, che analizza le
        temperatura massima di esposizione al piano parterre e primo, sono ininfluenti")
66
67
68
69  # ASET irraggiamento
70  soglia = 2.5
71  sensori = df.filter(regex='^I_').columns.tolist()
72  df_2 = df.loc[:, sensori]
73  df_3 = df_2[df_2 >= soglia]
74  conteggio = df_3.count().sum()
75  if conteggio > 0:
76      index_min = sys.maxsize
77      for colonna in df_3.columns:
78          index = df_3[colonna].first_valid_index()
79          if index != None:
80              if index < index_min:
81                  index_min = index
82      print("L'ASET risultato dall'analisi del modello del calore, analizzando gli
        output del massimo irraggiamento termico di esposizione degli occupanti, è:{:.2f}"
        .format(df['Time'][index_min] )+ "secondi")
83  else:
84      print("I valori risultanti dall'analisi del modello del calore, che analizza il
        massimo irraggiamento termico di esposizione degli occupanti, sono ininfluenti")
85
86
87
88  # ASET visibilità al piano secondo
89  soglia = 10
90  sensori = df.filter(regex='^V_P2').columns.tolist()
91  df_2 = df.loc[:, sensori]
92  df_2 = df_2.astype(float)
93  df_3 = df_2[df_2 <= soglia]
94  conteggio = df_3.count().sum()
95  if conteggio > 0:
96      index_min = sys.maxsize
97      for colonna in df_3.columns:
98          index = df_3[colonna].first_valid_index()
99          if index != None:
100             if index < index_min:
101                 index_min = index
102     print("L'ASET risultato dall'analisi del modello dello oscuramento della
        visibilità da fumo, analizzando gli output al piano secondo, è: {:.2f}".format(df[
        'Time'][index_min] )+ " secondi")
103 else:
104     print("I valori risultanti dall'analisi del modello dell'oscuramento della
        visibilità da fumo, che analizza la minima visibilità valutata al piano secondo,
        sono ininfluenti")
105
106
107 # ASET visibilità al piano parterre e primo
108 soglia = 10
109 sensori = df.filter(regex='^V(?!_P2)').columns.tolist()
110 df_2 = df.loc[:, sensori]
111 df_2 = df_2.astype(float)
112 df_3 = df_2[df_2 <= soglia]
113 conteggio = df_3.count().sum()
114 if conteggio > 0:
115     index_min = sys.maxsize
116     for colonna in df_3.columns:
117         index = df_3[colonna].first_valid_index()
118         if index != None:
119             if index < index_min:
120                 index_min = index
121     print("L'ASET risultato dall'analisi del modello dello oscuramento della
        visibilità da fumo, analizzando gli output al piano parterre e piano primo,
        è:{:.2f}".format(df['Time'][index_min] )+ "secondi")
122 else:
123     print("I valori risultanti dall'analisi del modello dell'oscuramento della
        visibilità da fumo, che analizza la minima visibilità al piano parterre e piano
```

```
        primo, sono ininfluenti")
124
125
126
127 # ASET FED piano secondo
128 soglia = 0.1
129 sensori = df.filter(regex='^F_P2').columns.tolist()
130 df_2 = df.loc[:, sensori]
131 df_3 = df_2[df_2 >= soglia]
132 conteggio = df_3.count().sum()
133 if conteggio > 0:
134     index_min = sys.maxsize
135     for colonna in df_3.columns:
136         index = df_3[colonna].first_valid_index()
137         if index != None:
138             if index < index_min:
139                 index_min = index
140     print("L'ASET risultato dall'analisi del modello dei gas tossici, analizzando gli
        output della FED al piano secondo (fractional effective dose), è:{:.2f}".format(df
        ['Time'][index_min] )+ "secondi")
141 else:
142     print("I valori risultanti dall'analisi del modello dei gas tossici, analizzando
        gli output della FED al piano secondo (fractional effective dose), sono
        ininfluenti")
143
144
145
146 # ASET FED piano parterre e primo
147 soglia = 0.1
148 sensori = df.filter(regex='^F_P1' and '^F_PT').columns.tolist()
149 df_2 = df.loc[:, sensori]
150 df_3 = df_2[df_2 >= soglia]
151 conteggio = df_3.count().sum()
152 if conteggio > 0:
153     index_min = sys.maxsize
154     for colonna in df_3.columns:
155         index = df_3[colonna].first_valid_index()
156         if index != None:
157             if index < index_min:
158                 index_min = index
159     print("L'ASET risultato dall'analisi del modello dei gas tossici, analizzando gli
        output della FED al piano parterre e primo (fractional effective dose), è:{:.2f}".
        format(df['Time'][index_min] )+ "secondi")
160 else:
161     print("I valori risultanti dall'analisi del modello dei gas tossici, analizzando
        gli output della FED al piano parterre e primo (fractional effective dose), sono
        ininfluenti")
162
163
164
165 # Creazione di un DataFrame vuoto per le medie di tutti gli intervalli poichè
    potrebbe esserci la necessità
166 # di calcolare la media negli intervalli
167 df_medie = pd.DataFrame(columns=dict(zip(df.columns, col_names)))
168 df_medie.rename(columns={'Time': 'Interval'}, inplace=True)
169 # Interpolazione dei risultati con un range di 10, data l'oscillazione dei valori,
    for i in range(0,x,y),
170 # x è la durata della simulazione, in y deve essere sostituito l'intervallo di tempo
    su cui fare la media
171 for i in range(0, 1200, 10):
172     df_intervallo = df[(df['Time'] > i) & (df['Time'] < i + 10)]
173     df_media = df_intervallo.mean()
174     interval = str(i)
175     df_temp = pd.concat([pd.DataFrame([interval]), df_media.to_frame().T], axis=1)
176     df_temp.columns = ['Interval'] + df_media.index.tolist()
177     df_medie = pd.concat([df_medie, df_temp])
178 df_medie = df_medie.drop('Time', axis=1)
179 df_medie = df_medie.reset_index(drop=True)
180 df_medie
181
182
183
```

```
# Stampa sensori di temperatura
y_plot = df_medie.filter(regex='^T').columns.tolist()
ax = df_medie.plot(x='Interval', y=y_plot, kind='line', grid=True, figsize=(10,10))
# Set origine
ax.set_xlim(left=0)
ax.set_ylim(bottom=19.9)
# Set caratteristiche e posizione legenda
legend = ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
# Titoli
legend.set_title('Legenda sensori\n di temperatura')
ax.set_xlabel('Tempo [s]')
ax.set_ylabel('Temperatura [°C]')
plt.show()



# Stampa sensori di visibilità del Piano 2°
y_plot = df_medie.filter(regex='^V_P2').columns.tolist()
ax = df_medie.plot(x='Interval', y=y_plot, kind='line', grid=True, figsize=(10,10))
ax.set_xlim(left=0)
ax.set_ylim(bottom=0)
legend = ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
legend.set_title('Legenda sensori\n    di visibilità')
ax.set_xlabel('Tempo [s]')
ax.set_ylabel('Visibilità [m]')
plt.show()



# Stampa sensori di visibilità del piano Terra e del 1° piano
y_plot = df_medie.filter(regex='^V(?!_P2)').columns.tolist()
ax = df_medie.plot(x='Interval', y=y_plot, kind='line', grid=True, figsize=(10,10))
ax.set_xlim(left=0)
ax.set_ylim(bottom=0)
legend = ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
legend.set_title('Legenda sensori\n    di visibilità')
ax.set_xlabel('Tempo [s]')
ax.set_ylabel('Visibilità [m]')
plt.show()



# Stampa sensori di irraggiamento
y_plot = df_medie.filter(regex='^I_').columns.tolist()
ax = df_medie.plot(x='Interval', y=y_plot, kind='line',grid=True, figsize=(10,10))
ax.set_xlim(left=0)
ax.set_ylim(bottom=0)
legend = ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
legend.set_title('Legenda sensori\ndi irraggiamento')
ax.set_xlabel('Tempo [s]')
ax.set_ylabel('Irraggiamento [kW/m²]')
plt.show()



# Stampa sensori di gas tossici
y_plot = df_medie.filter(regex='^F_').columns.tolist()
ax = df_medie.plot(x='Interval', y=y_plot, kind='line', grid=True, figsize=(10,10))
ax.set_xlim(left=0)
ax.set_ylim(bottom=0)
legend = ax.legend(loc='upper right', bbox_to_anchor=(1.2, 1))
legend.set_title('Legenda sensori\n       dei gas tossici')
ax.set_xlabel('Tempo [s]')
ax.set_ylabel('Gas tossici [mol/mol]')
plt.show()
```