Eugenio Principi - s236654
Arya Houshmand - s247275
Leonardo Perugini - s249807

## Esercizi 1.1 - 1.2 - 1.3

Nei primi tre esercizi ci siamo occupati di creare dei servizi web che simulano una conversione di temperatura avendo in input i seguenti tre parametri: 1) il valore numerico da convertire; 2) l'unità di misura di partenza; 3) l'unità di misura in cui si vuole la conversione.

Negli esercizi 1.1 e 1.2 si utilizza una richiesta HTTP GET, in particolare nel secondo caso si chiede di esporre per intero la URL.

```python
import cherrypy
import json

def CtoK (x):
    return round(x + 273.15,2)

def KtoC(x):
    return round(x - 273.15,2)

def CtoF(x):
    return round(x * 9/5 + 32,2)

def FtoC(x):
    return round((x - 32) * 5/9,2)

def FtoK(x):
    return round((x - 32) * 5/9 + 273.15)

def KtoF(x):
    return round((x -273.15) * 9/5 + 32)
```

In tutti gli esercizi si sono definite queste funzioni per effettuare il calcolo attraverso che converte la temperatura da un' unità di misura ad un' altra e inoltre rendono il codice più compatto e modulare.

Nell' esercizio 1.3 si chiede invece di convertire una lista di valori di temperature, e non più un singolo valore. Inoltre in questo caso viene usata una HTTP PUT che riceve i valori di input tramite un file JSON e verrà restituito il risultato con un file JSON con la stessa struttura.
Qui di seguito viene presentato un pezzo di codice dell'esercizio 1.3:

```
def PUT(self, **params):
    data = cherrypy.request.body.read()
    json_data = json.loads(data.decode('utf-8'))

    errore = "Unita' di misura errata"

    x = 'originalUnit'
    if(json_data[x] != 'C' and json_data[x] != 'K' and json_data[x] != 'F'):
        return errore

    y = 'targetUnit'
    if(json_data[y] != 'C' and json_data[y] != 'K' and json_data[y] != 'F'):
        return errore

    newValues = []

    if(json_data[x] == 'C'):
        if(json_data[y] == 'K'):
            for j in json_data["values"]:
                newValues.append(CtoK(int(j)))
        else:
            for j in json_data["values"]:
                newValues.append(CtoF(int(j)))


    if(json_data[x] == 'K'):
        if(json_data[y] == 'C'):
            for j in json_data["values"]:
                newValues.append(KtoC(int(j)))
        else:
            for j in json_data["values"]:
                newValues.append(KtoF(int(j)))


    if(json_data[x] == 'F'):
        if(json_data[y] == 'C'):
            for j in json_data["values"]:
                newValues.append(FtoC(int(j)))
        else:
            for j in json_data["values"]:
                newValues.append(FtoK(int(j)))

    conversion = { json_data[x] : json_data['values'], json_data[y] : newValues }
    return json.dumps(conversion)
```

## Esercizio 1.4

In tale esercizio invece si sviluppa un web service REST per esporre la *“freeboard”.* Attraverso un metodo HTTP GET si ottiene l’ *index.html*, mentre attraverso delle HTTP POST vengono salvate le configurazioni della dashboard tramite un file JSON.
Anche in questo caso riportiamo qui di seguito uno snippet del codice:

```
class WebService():

    exposed = True
    def GET(self, *uri, **params):
        return open("index.html","r").read()

    def POST(self, *uri, **params):
        if uri[0] == "saveDashboard":
            path = "dashboard/"
            with open(path + "dashboard.json", "w") as file:
                file.write(params['json_string'])
```