Python: lavorare con le date

In questo articolo vedremo un esempio pratico di operazioni sulle date in Python.

Realizzeremo una semplice applicazione da riga di comando che riceverà in input la data di nascita dell'utente e restituirà il nome del suo segno zodiacale.

I segni zodiacali sono dodici e ciascuno di loro copre un determinato arco temporale nel corso dell'anno. Per rappresentarli useremo un file JSON contenente un array di oggetti così strutturati:

```
[
    {
        "name": "Nome Segno",
        "date": {
            "from": [giorno, mese],
            "to": [giorno, mese]
        }
    }
]
```

`giorno` e `mese` sono numeri interi. Una caratteristica del costruttore della classe `date` di Python è appunto quella di accettare solo numeri interi per dati come l'anno, il mese e il giorno. `from` rappresenta la data di inizio del periodo di un segno e `to` il suo termine. L'anno è ininfluente.

Dobbiamo accettare solo date in formato ISO, ossia `YYYY-mm-dd` e verificare che anno, mese e giorno siano validi. A tale scopo definiamo come prima cosa una funzione di validazione.

```
import json
import re
import sys
from datetime import date, datetime


def is_valid_date_str(date_str):
    reg = re.compile(r'^\d{4}-\d{2}-\d{2}$')
    match = reg.search(date_str)
    if not match:
        return False
    try:
        dt = date.fromisoformat(date_str)
        year = dt.year
        now = datetime.now()
```

```
        if year >= now.year:
            return False
        month = dt.month
        if month > 12 or month < 1:
            return False
        day = dt.day
        if day > 31 or day < 1:
            return False
    except ValueError:
        return False
    return True
```

Il primo step è verificare il formato della data usando una semplice espressione regolare. Usiamo `search()` per effettuare un confronto sull'intera stringa di input. A questo punto se il formato è valido, corroboriamo la nostra routine ottenendo un oggetto di tipo `date` a partire dal formato ISO fornito usando il metodo `fromisoformat()`. Inoltre procediamo a verificare che anno, mese e giorno ricadano in un intervallo valido.

Ora possiamo definire una funzione per ottenere la lista di dizionari dal file JSON.

```
def get_signs_list(filename=None):
    signs = []
    if filename is None:
        return signs
    with open(filename, 'r') as f:
        signs = json.load(f)
    return signs
```

Sappiamo che i dati rilevanti sono il giorno e il mese della data di nascita dell'utente e che tali dati vanno confrontati con i valori contenuti nelle liste `from` e `to` di ciascun dizionario. Dobbiamo quindi operare un confronto tra tre oggetti `date` partendo dalla trasformazione della stringa nel formato `YYYY-mm-dd` della data di nascita in un oggetto data di Python. Quest'operazione può essere effettuata tramite il metodo `datetime.strptime()` che accetta come primo argomento la stringa data e come secondo argomento il formato atteso della data di input.

```
def find_sign_by_birth_date(birthdate=None):
    if birthdate is None:
        return None
    if not is_valid_date_str(birthdate):
        return None
    signs_list = get_signs_list('./zodiac-signs.json')
    format_date = '%Y-%m-%d'
    dt = datetime.strptime(birthdate, format_date)
    now = datetime.now()
    day = dt.day
    month = dt.month
    reference_date = date(now.year, month, day)
```

```
    sign_name = ''
    for sign in signs_list:
        from_day, from_month = sign['date']['from']
        to_day, to_month = sign['date']['to']
        from_date = date(now.year, from_month, from_day)
        to_date = date(now.year, to_month, to_day)

        if reference_date >= from_date and reference_date <= to_date:
            sign_name = sign['name']
            break

    return sign_name
```

Se la data di nascita ricade nel range stabilito, ossia se è maggiore o uguale della data iniziale del periodo e se al contempo è minore o uguale della data finale del periodo, restituiamo il nome del segno zodiacale dell'utente. Ricordiamo ancora che l'anno è ininfluente ma è necessario specificarlo per creare un oggetto `date` di Python.

Possiamo usare il nostro codice nel seguente modo:

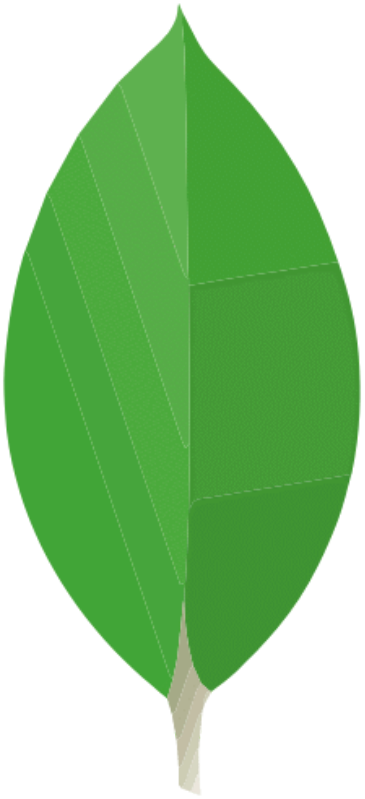
```
def main():
    birth_date = input('Insert your birth date (YYYY-mm-dd)')
    sign_found = find_sign_by_birth_date(birth_date)
    if sign_found is None:
        print('Invalid birth date')
        sys.exit(1)
    print(f'Your sign is: {sign_found}')
    sys.exit(0)


if __name__ == '__main__':
    main()
```

Applicazioni Correlate

-

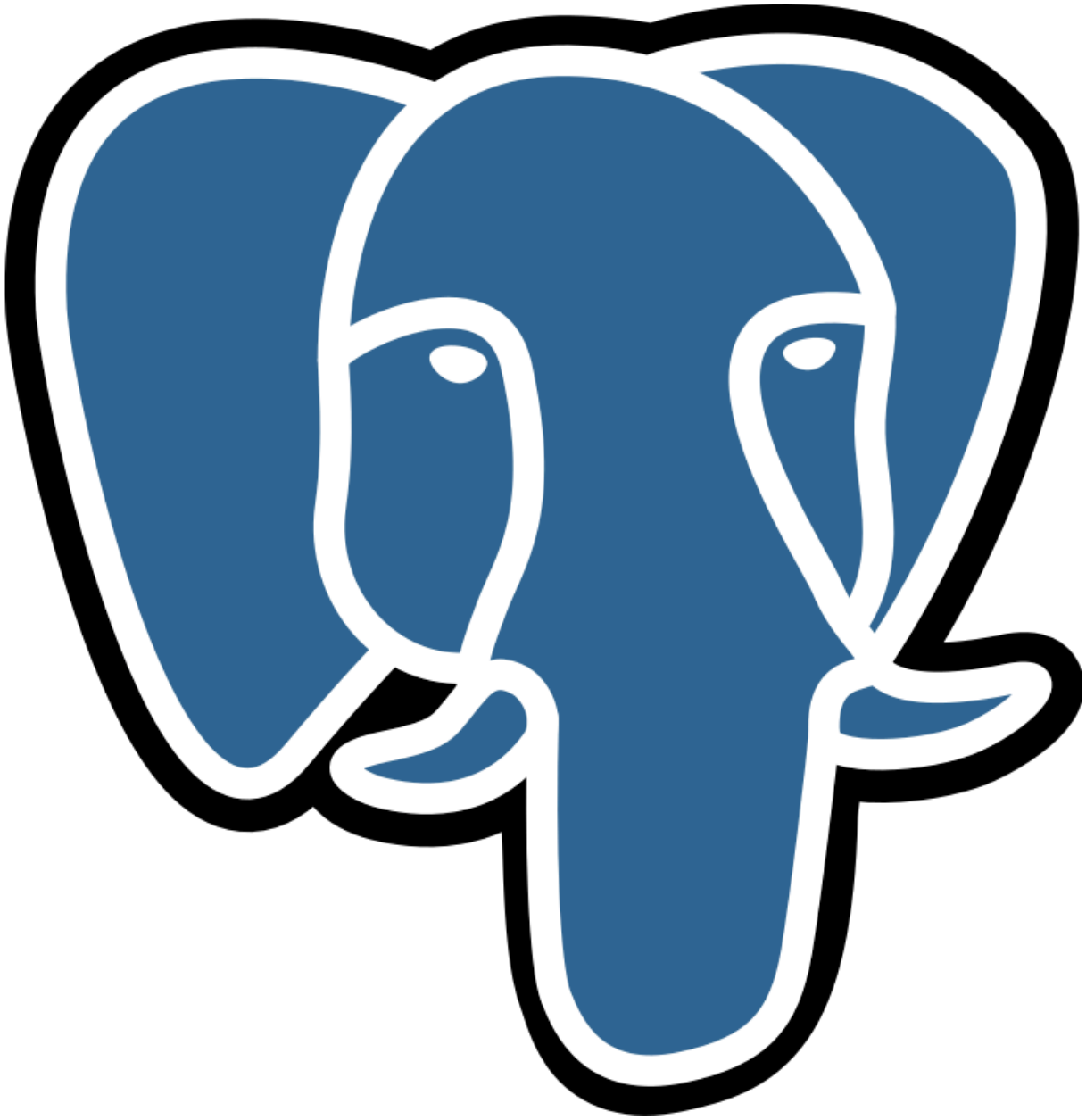
Python Placeholder Image

Applicazione sviluppata in Python con Flask per la creazione di immagini segnaposto.
DockerDocker ComposePythonFlaskJavaScript



Python MongoDB App

Applicazione basata su MongoDB e sviluppata in Python e Flask.
DockerDocker ComposePythonFlaskMongoDB



Python PostgreSQL App

Applicazione basata su PostgreSQL con Python e Flask.
DockerDocker ComposePythonFlask