Kullanıcıdan bir NFA mödülü girilmesi ve bu girilen modüle göre bunu DFA otomatasına dönüştürülmesini Python dilini kullanarak aşağıda verdiğim kodu yazdım.

```
import json
from collections import OrderedDict

def get_nfa_from_user():
    nfa = {}
    # Kullanıcıdan NFA'nın durum sayısını al
    nfa["states"] = int(input("Durum sayisini girin: "))
    # Kullanıcıdan NFA'nın alfabe harflerini al
    nfa["letters"] = input("Alfabe harflerini virgülle ayirarak girin:
").split(',')
    # Kullanıcıdan NFA'nın başlangıç durumunu al
    nfa["start"] = input("Başlangiç durumunu girin: ")
    # Kullanıcıdan NFA'nın geçiş fonksiyonlarını al
    nfa["t_func"] = []
    print("Geçiş fonksiyonlarini girin (format: durumu,harf,durumlar).Bitirmek
için 'q' girin.")
    while True:
        transition = input("Geçiş fonksiyonu: ")
        if transition.lower() == 'q':
            break
        from_state, symbol, to_states = transition.split(',')
        to_states = to_states.split()
        nfa["t_func"].append([from_state, symbol, to_states])
    # Kullanıcıdan NFA'nın son durumlarını al
    nfa["final"] = input("Son durumlari virgülle ayirarak girin:").split(',')
    return nfa
def nfa_to_dfa(data):
    dfa_states = 2 ** data["states"]
    dfa_letters = data["letters"]
    dfa_start = data["start"]
    dfa_t_func = []
    dfa_final = []
    dfa_list = []
    q = []
    q.append((dfa_start,))
    nfa_transitions = {}
    dfa_transitions = {}
    for transition in data["t_func"]:
        nfa_transitions[(transition[0], transition[1])] = transition[2]
    for in_state in q:
        for symbol in dfa_letters:
            if len(in_state) == 1 and (in_state[0], symbol) in
nfa_transitions:
```

```
                dfa_transitions[(in_state, symbol)] =
nfa_transitions[(in_state[0], symbol)]
                if tuple(dfa_transitions[(in_state, symbol)]) not in q:
                    q.append(tuple(dfa_transitions[(in_state, symbol)]))
            else:
                dest = []
                f_dest = []
                for n_state in in_state:
                    if (n_state, symbol) in nfa_transitions and
nfa_transitions[(n_state, symbol)] not in dest:
                        dest.append(nfa_transitions[(n_state, symbol)])
                if dest:
                    for d in dest:
                        for value in d:
                            if value not in f_dest:
                                f_dest.append(value)
                    dfa_transitions[(in_state, symbol)] = f_dest
                    if tuple(f_dest) not in q:
                        q.append(tuple(f_dest))
    for key, value in dfa_transitions.items():
        temp_list = [[key[0], key[1], value]]
        dfa_t_func.extend(temp_list)
    for q_state in q:
        for f_state in data["final"]:
            if f_state in q_state:
                dfa_final.append(q_state)
    dfa = OrderedDict()
    dfa["states"] = dfa_states
    dfa["letters"] = dfa_letters
    dfa["t_func"] = dfa_t_func
    dfa["start"] = dfa_start
    dfa["final"] = dfa_final
    return dfa
nfa_data = get_nfa_from_user()
dfa_data = nfa_to_dfa(nfa_data)
with open('output.json', 'w') as output_file:
    json.dump(dfa_data, output_file, separators=(',', ':'))
print("DFA, output.json dosyasina yazildi.")
```

Şimdi bu kodu ne iş yapar sırayla açıklamak isterim.

İlk olarak kodumun bu kısmını ele alırsam

```python
def get_nfa_from_user():
    nfa = {}

    # Kullanıcıdan NFA'nın durum sayısını al
    nfa["states"] = int(input("Durum sayisini girin: "))

    # Kullanıcıdan NFA'nın alfabe harflerini al
    nfa["letters"] = input("Alfabe harflerini virgülle ayirarak girin: ").split(',')

    # Kullanıcıdan NFA'nın başlangıç durumunu al
    nfa["start"] = input("Başlangiç durumunu girin: ")

    # Kullanıcıdan NFA'nın geçiş fonksiyonlarını al
    nfa["t_func"] = []
    print("Geçiş fonksiyonlarini girin (format: durumu,harf,durumlar). Bitirmek için 'q' girin.")
    while True:
        transition = input("Geçiş fonksiyonu: ")
        if transition.lower() == 'q':
            break
        from_state, symbol, to_states = transition.split(',')
        to_states = to_states.split()
        nfa["t_func"].append([from_state, symbol, to_states])

    # Kullanıcıdan NFA'nın son durumlarını al
    nfa["final"] = input("Son durumlari virgülle ayirarak girin: ").split(',')

    return nfa
```

Bu kısımda bir fonksiyon oluşturdum bu fonksiyonumun görevi kullanıcıdan bir NFA otomatasının tüm bileşenlerini girmesini ister.
İlk olarak kullanıcıya NFA'nın durum sayısını girmesi istenir ve bu sayı nfa sözlüğüne "states" anahtarı altında atanır.
İkinci olarak kullanıcıya NFA'nın kullandığı alfabe harflerini virgülle ayırarak girmesi istenir. input() fonksiyonuyla alınan girdi, split(',') metoduyla virgüle göre ayrılarak bir liste haline getirilir ve bu liste nfa sözlüğünde "letters" anahtarı altında saklanır.
Üçüncü olarak kullanıcıya NFA'nın başlangıç durumunu girmesi istenir ve bu değer nfa sözlüğüne "start" anahtarı altında atanır.
Dördüncü adımımız kullanıcıya geçiş fonksiyonlarını girmesi istenir. Kullanıcı geçişi tanımlamak için "durumu,harf,durumlar" formatında giriş yapar. Her bir geçiş, bir listenin içinde saklanır ve bu listeler nfa sözlüğünde "t_func" anahtarı altında bir liste olarak saklanır. Kullanıcı "q" tuşuna basana kadar bu işlem devam eder
Son olarak ise kullanıcıya NFA'nın son durumlarını girmesi istenir. Kullanıcı bu durumları virgülle ayırarak girecektir. Girdi alındıktan sonra, virgüle göre ayrılan değerler bir liste haline getirilir ve bu liste nfa sözlüğünde "final" anahtarı altında saklanır.
Bu adımlar tamamlandıktan sonra, nfa sözlüğü, kullanıcının girdilerine dayalı olarak NFA'nın tüm bileşenlerini içerecektir ve bu sözlük return anahtar kelimesiyle fonksiyon tarafından döndürülür.

İkinci olarak kodumun nfa dan dfa kısmına dönüşüm fonksiyonumu ele alalım

```python
def nfa_to_dfa(data):
    dfa_states = 2 ** data["states"]
    dfa_letters = data["letters"]
    dfa_start = data["start"]
    dfa_t_func = []
    dfa_final = []
    q = []

    q.append((dfa_start,))

    nfa_transitions = {}
    dfa_transitions = {}

    for transition in data["t_func"]:
        nfa_transitions[(transition[0], transition[1])] = transition[2]

    for in_state in q:
        for symbol in dfa_letters:
            if len(in_state) == 1 and (in_state[0], symbol) in nfa_transitions:
                dfa_transitions[(in_state, symbol)] = nfa_transitions[(in_state[0], symbol)]

                if tuple(dfa_transitions[(in_state, symbol)]) not in q:
                    q.append(tuple(dfa_transitions[(in_state, symbol)]))
            else:
                dest = []
                f_dest = []

                for n_state in in_state:
                    if (n_state, symbol) in nfa_transitions and nfa_transitions[(n_state, symbol)] not in dest:
                        dest.append(nfa_transitions[(n_state, symbol)])

                if dest:
                    for d in dest:
                        for value in d:
                            if value not in f_dest:
                                f_dest.append(value)

                    dfa_transitions[(in_state, symbol)] = f_dest

                    if tuple(f_dest) not in q:
                        q.append(tuple(f_dest))

    for key, value in dfa_transitions.items():
        temp_list = [[key[0], key[1], value]]
        dfa_t_func.extend(temp_list)

    for q_state in q:
        for f_state in data["final"]:
            if f_state in q_state:
                dfa_final.append(q_state)

    dfa = OrderedDict()
    dfa["states"] = dfa_states
    dfa["letters"] = dfa_letters
    dfa["t_func"] = dfa_t_func
    dfa["start"] = dfa_start
    dfa["final"] = dfa_final
    return dfa
```

İlk olarak DFA durumlarını hesaplanır. data adlı bir NFA sözlüğü alır ve bu NFA'dan türetilen DFA'nın toplam durum sayısını hesaplar. Her bir NFA durumu, 2'nin kuvveti kadar olacak şekilde DFA durumlarını temsil eder.
Bu işlemden sonra NFA'nın kullandığı alfabe harflerini kopyalar. Bir sonraki adımda da aynı mantıkla NFA'nın başlangıç durumunu kopyalar.
Sonra DFA Geçiş Fonksiyonları ve Hedef Durumları Hesaplanır :
*** q adında bir liste oluşturur. Bu liste, dönüşüm sırasında oluşturulan DFA durumlarını tutar.**
***Başlangıç durumu q listesine eklenir.**

***NFA'dan DFA'ya geçişler ve hedef durumlar hesaplanır.**
***Her bir DFA durumu için geçişler ve hedef durumlar dfa_transitions adında bir sözlükte saklanır.**
Bu işlemler yapıldıktan sonra. Her DFA durumu için, eğer bu durum NFA'nın bir son durumunu içeriyorsa, bu durum DFA'nın son durumları listesine eklenir.
En son olarakta oluşturulan tüm bilgiler kullanılarak DFA sözlüğü oluşturulur ve döndürülür.

Son olarak kodumun bu kısmını ele alırsam

```
nfa_data = get_nfa_from_user()
dfa_data = nfa_to_dfa(nfa_data)

with open('output.json', 'w') as output_file:
    json.dump(dfa_data, output_file, separators=(',', ':'))

print("DFA, output.json dosyasina yazildi.")
```

Son olarak, elde edilen DFA'yı JSON formatında output.json dosyasına yazar. NFA'dan DFA'ya geçişi adım adım gerçekleştirir ve sonuçları kullanıcıya json dosyasında gösterir.

Örnek olarak bir kullanıcı yazdığım koda veri girişi yapmış olsun

->Durum Sayısı: 4
-> Alfabe Harfleri: a, b
-> Başlangıç Durumu: q0
-> Geçiş Fonksiyonları: q0, a -> q0, q1
q0, b -> q0,q3
q1, a -> q2
q2,a -> q2
q2,b -> q2
q3,b -> q2
-> Son Durumlar: q2

Bu örnek giriş üzerinden NFA dan DFA' ya dönüşüm otomatamı

- Tablo formatında
- Grafik formatında
- Formal dil biçiminde

Bu üç formatta gösterimi

1) FORMAT DİLİ BİÇİMİNDE

INPUT NFA

Q = { {q0}, {q1}, {q2}, {q3}}
Σ = {a, b}
q0 = {q0}
F = { {q2} }
δ({q0}, a) = {q0, q1},
δ({q0}, b) = {q0, q3},
δ({q1}, a) = {q2} ,
δ({q2}, a) = {q2},
δ({q2}, b) = {q2},
δ({q3}, b) = {q2}

OUTPUT DFA

Q = { {q0}, {q1}, {q2}, {q3}}
Σ = {a, b}
q0 = {q0}
F = { {q2} }
δ({q0}, a) = { q1} ,
δ({q0}, b) = {q3},
δ({q1}, a) = {q2} ,
δ({q1}, b) = {q3},
δ({q2}, a) = {q2},
δ({q2}, b) = {q2},
δ({q3}, a) = {q0},
δ({q3}, b) = {q2}

2) TABLO FORMATINDA

INPUTNFA

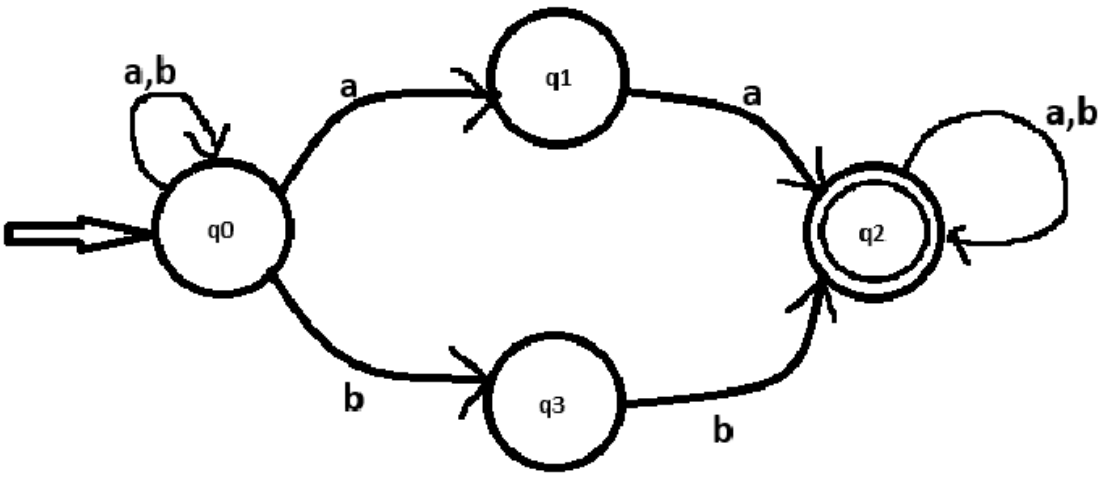
DURUMLAR	a	b
->q0	{q0,q1}	{q0,q3}
q1	{q2}	{}
q2	{q2}	{q2}
q3	{}	{q2}

OUTPUT DFA

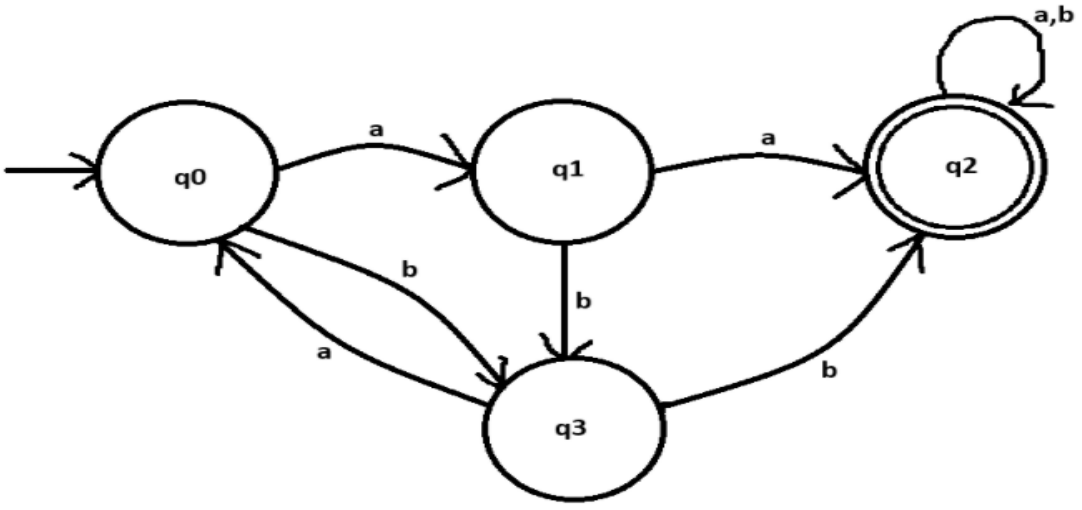
DURUMLAR	a	b
q0	q1	q3
q1	q2	q3
q2	q2	q2
q3	q0	q2

3) GRAFİK FORMATINDA

INPUT NFA



OUTPUT DFA



NFA otomatamızın DFA otomatamıza dönüşümünü Formal dil biçiminde , Tablo ve Grafik formatında göstermiş oldum.

Bu örnekteki NFA otomatamı koduma girdiğimde burada çıkan sonuçlarla aynı bir DFA otomatasını elde etmiş oluruz.