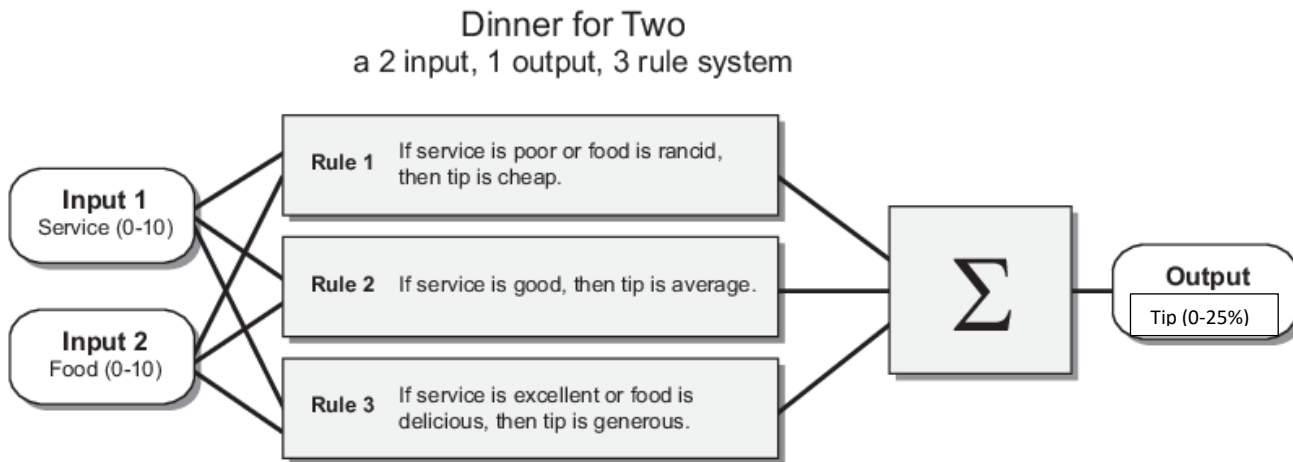


می‌خواهیم مسأله زیر را در python حل کنیم:



کتابخانه‌های زیادی برای استفاده از منطق فازی در python وجود دارند، البته هیچ کدام از آنها جزء کتابخانه‌های استاندارد نیستند. کتابخانه‌هایی مانند `fuzzylogic`, `skfuzzy`, `fuzzylab`, `simpful`.

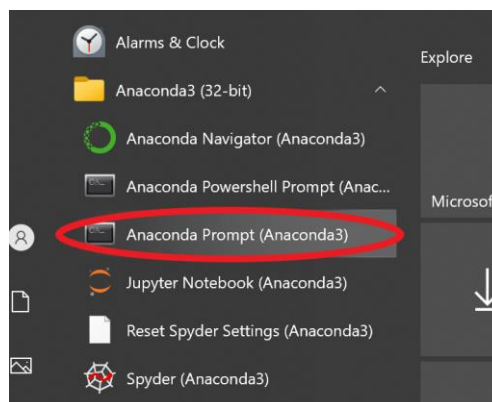
این کتابخانه‌ها همگی در PyPi قرار دارند. PyPi یا Python Package Index، مخزنی نرم‌افزاری برای برنامه‌نویسی پایتون است که امکان استفاده از نرم‌افزارهایی که توسط جامعه پایتون نوشته شده است و توسط آنها در این مخزن به اشتراک گذاشته شده است را فراهم می‌کند.

هر بسته نرم‌افزاری که در این مخزن قرار دارد را باید ابتدا در محیط برنامه‌نویسی خود نصب کنیم.

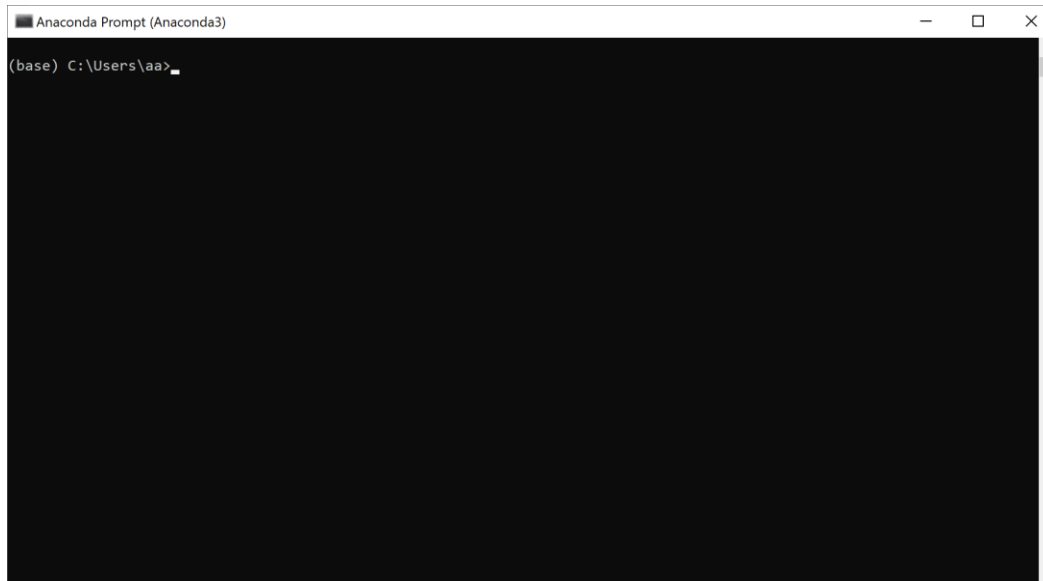
در ادامه برای برنامه‌نویسی این مسأله فازی از کتابخانه `scikit-fuzzy` (skfuzzy) استفاده می‌کنیم.

الف) مراحل نصب کتابخانه `scikit-fuzzy`

برای کار با پایتون باید برنامه Anaconda را نصب کرده باشید. یکی از قسمت‌های این برنامه Anaconda Prompt است که در شکل زیر مشاهده می‌کنید:



بر روی آن کلیک کنید تا صفحه مربوط به آن باز شود:



در اینجا برای نصب هر package موردنظر که در PyPi وجود دارد، دستور

`pip install <package-name>`

را می‌نویسیم. (سیستم باید به اینترنت متصل باشد.)

پس باید دستور زیر را بنویسیم:

`pip install scikit-fuzzy`



و دکمه **enter** را فشار دهیم تا دانلود و نصب برنامه در محیط پایتون انجام شود. از این مرحله به بعد می‌توانیم از این کتابخانه در برنامه‌های پایتون خود استفاده کنیم.

ب) در ادامه به توضیح برنامه مربوط به سیستم فازی اشاره شده در محیط Jupyter Notebook می‌پردازیم.

skfuzzy بر اساس کتابخانه numpy پایتون عمل می‌کند، به همین دلیل باید آن را هم در برنامه import کنیم. همچنین control را import می‌کنیم تا برنامه را به‌عنوان یک سیستم کنترلی بنویسیم.

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl
```

حالا دو متغیر ورودی و یک متغیر خروجی و بازه آنها را مشخص می‌کنیم (اولی کیفیت غذا، دومی کیفیت خدمات، و سومی میزان انعام را مشخص می‌کند):

```
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
```

اولی در بازه 0 تا 10، دومی در بازه 0 تا 10، امتیازاتی است که از کاربر گرفته می‌شود. سومی در بازه 0 تا 25، درصدی از قیمت غذا است که باید به عنوان خروجی برگردانده شود.

در مرحله بعد، توابع عضویت را مشخص می‌کنیم، اینجا همه را به صورت مثلثی تعریف کرده‌ایم (با استفاده از تابع trimf):

```
quality['low'] = fuzz.trimf(quality.universe, [0, 0, 5])
quality['medium'] = fuzz.trimf(quality.universe, [0, 5, 10])
quality['high'] = fuzz.trimf(quality.universe, [5, 10, 10])

service['low'] = fuzz.trimf(service.universe, [0, 0, 5])
service['medium'] = fuzz.trimf(service.universe, [0, 5, 10])
service['high'] = fuzz.trimf(service.universe, [5, 10, 10])

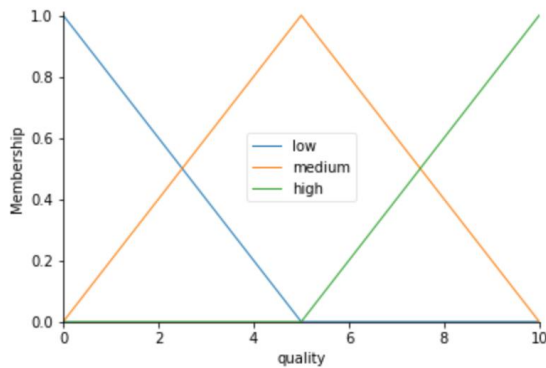
tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])
```

نکته: برای تولید توابع عضویت، متدهای مختلفی در این کتابخانه تعریف شده‌اند: trimf (تابع عضویت مثلثی)، trapmf (تابع عضویت ذوزنقه‌ای)، sigmf (تابع عضویت سیگموئید) و

با دستورات زیر می‌توانیم نمودارهای آنها را مشاهده کنیم:

```
quality.view()
service.view()
tip.view()
```

مثلا خروجی مربوط به quality:



در مرحله بعد، rule ها را تعریف می کنیم:

این سه قانون را داشتیم:

1. If the food is poor OR the service is poor, then the tip will be low
2. If the service is average, then the tip will be medium
3. If the food is good OR the service is good, then the tip will be high.

که کد آنها معادل زیر می شود:

```
rule1 = ctrl.Rule(quality['low'] | service['low'], tip['low'])
rule2 = ctrl.Rule(service['medium'], tip['medium'])
rule3 = ctrl.Rule(service['high'] | quality['high'], tip['high'])
```

نکته) اگر جایی، بین قوانین به جای OR، AND داشته باشیم، به جای | از & استفاده می کنیم.

حالا سیستم فازی خود را بر مبنای این سه قانون تعریف می کنیم:

```
tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
```

سپس امکان شبیه سازی این سیستم فازی را به وسیله دستور زیر فراهم می کنیم:

```
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
```

حالا ورودی های سیستم شبیه ساز را وارد می کنیم:

```
tipping.input['quality'] = 4.5
tipping.input['service'] = 9.8
```

کارهای لازم انجام شد و سیستم را اجرا می کنیم:

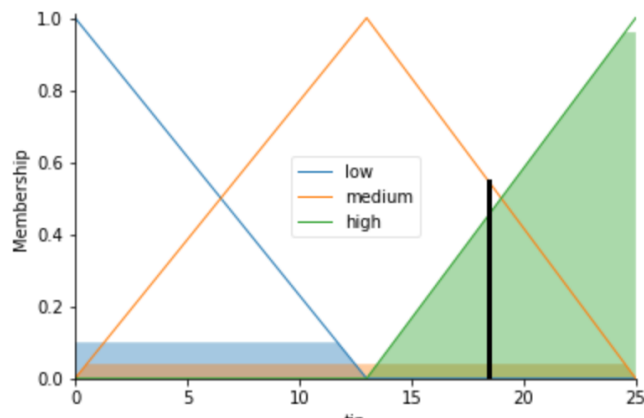
```
tipping.compute()
```

خروجی را با دستورات زیر می‌توانیم مشاهده کنیم:

```
print(tipping.output['tip'])
tip.view(sim=tipping)
```

خروجی:

18.431841697176573



کد کامل آن در Jupyter:

Cell 1:

```
import numpy as np
import skfuzzy as fuzz
from skfuzzy import control as ctrl

# New Antecedent/Consequent objects hold universe variables and membership
# functions
quality = ctrl.Antecedent(np.arange(0, 11, 1), 'quality')
service = ctrl.Antecedent(np.arange(0, 11, 1), 'service')
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')

# Custom membership functions can be built interactively with a familiar,
# Pythonic API
```

```

quality['low'] = fuzz.trimf(quality.universe, [0, 0, 5])
quality['medium'] = fuzz.trimf(quality.universe, [0, 5, 10])
quality['high'] = fuzz.trimf(quality.universe, [5, 10, 10])
service['low'] = fuzz.trimf(service.universe, [0, 0, 5])
service['medium'] = fuzz.trimf(service.universe, [0, 5, 10])
service['high'] = fuzz.trimf(service.universe, [5, 10, 10])

tip['low'] = fuzz.trimf(tip.universe, [0, 0, 13])
tip['medium'] = fuzz.trimf(tip.universe, [0, 13, 25])
tip['high'] = fuzz.trimf(tip.universe, [13, 25, 25])
# You can see how these look with .view()
quality.view()
service.view()
tip.view()

```

cell 2:

```

# rule defenition
rule1 = ctrl.Rule(quality['poor'] | service['poor'], tip['low'])
rule2 = ctrl.Rule(service['average'], tip['medium'])
rule3 = ctrl.Rule(service['good'] | quality['good'], tip['high'])

```

cell 3:

```

tipping_ctrl = ctrl.ControlSystem([rule1, rule2, rule3])
tipping = ctrl.ControlSystemSimulation(tipping_ctrl)
# Pass inputs to the ControlSystem using Antecedent labels with Pythonic API
# Note: if you like passing many inputs all at once, use .inputs(dict_of_data)
tipping.input['quality'] = 4.5
tipping.input['service'] = 9.8
# Crunch the numbers
tipping.compute()

```

```
print tipping.output['tip']
tip.view(sim=tipping)
```

چند نکته:

نکته اول) قبلا و در متن درس گفته بودیم، که سیستم‌های فازی موارد مختلفی دارند که باید درباره آنها تصمیم‌گیری شود: آن موارد عبارت بودند از: T-norm, S-norm, Implication, Aggregation, Fuzzification و Defuzzification.

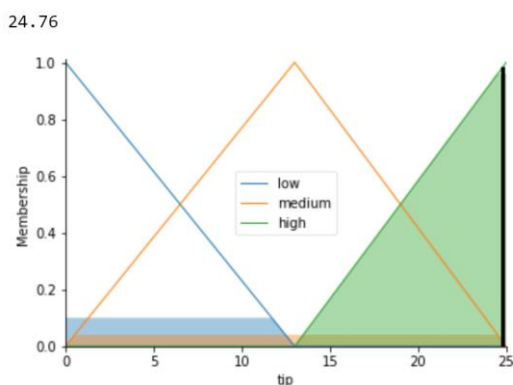
اما در skfuzzy مانند بسیاری دیگر از کتابخانه‌های فازی در زبان‌های مختلف، همه چیز بر اساس inference ممدانی مشخص شده است و تنها می‌توانید روش Defuzzification را تغییر دهید.

روش‌های Defuzzification تعریف شده که حالت پیش‌فرض آنها که کد نوشته شده در بالا هم بر اساس همان اجرا شد، centroid است، عبارت هستند از: lom(max of maximum), som(min of maximum), bisector, mom(mean of maximum) (of maximum)

برای تغییر روش دیفازی، دقیقا بعد از تعریف متغیر consequent، آن را مشخص می‌کنیم. در اینجا این متغیر tip است و یک خط کد به شکل زیر به برنامه اضافه شده است.

```
tip = ctrl.Consequent(np.arange(0, 26, 1), 'tip')
tip.defuzzify_method="mom"
```

این تنها تغییری است که در برنامه ایجاد شده است، و حالا با همان ورودی‌ها، خروجی به شکل زیر تغییر می‌کند:



نکته دوم) skfuzzy امکان تعریف توابع عضویت مثلثی به صورت خودکار را هم فراهم می‌کند. مثلا می‌توانید به جای تعریف توابع عضویت quality دستور زیر را بنویسید:

```
quality.automf(3)
```

که 3 تابع عضویت تعریف می‌کند، به جای 3، امکان گذاشتن، عددهای 5 و 7 نیز وجود دارد.

نکته 3)

`dir()` تابعی در `python` است که لیست تمام ویژگی‌ها و متدهای هر نوع `object` پایتونی را برمی‌گرداند. با نوشتن دستور زیر، قسمتی از خروجی آن را در ادامه مشاهده می‌کنید:

```
dir(fuzz)
```

```
['_INPLACE_MSG',  
 '_STANDARD_MSG',  
 '__SKFUZZY_SETUP__',  
 '__all__',  
 '__builtins__',  
 '__cached__',  
 '__doc__',  
 '__file__',  
 '__loader__',  
 '__name__',  
 '__package__',  
 '__path__',  
 '__spec__',  
 '__version__',  
 '_cluster',  
 '_defuzz',  
 '_filters',  
 '_fuzzymath',  
 '_image',  
 '_intervals',  
 .
```

مثلاً یکی از متدها `fuzzy_add` بود. با نوشتن دستور `fuzz.fuzzy_add` و سپس کلیک دکمه‌های `shift+Tab+Tab` خروجی زیر که توضیحی درباره عملکرد، ورودی‌ها و خروجی این تابع است به ما داده می‌شود.

```
fuzz.fuzzy_add|  
  
Signature: fuzz.fuzzy_add(x, a, y, b)  
Docstring:  
Add fuzzy set ``a`` to fuzzy set ``b``.  
  
Parameters  
-----  
x : 1d array, length N  
    Universe variable for fuzzy set ``a``.  
a : 1d array, length N  
    Fuzzy set for universe ``x``.
```

نکته 4) دیدیم که فقط توانستیم روش دیفازی را تغییر دهیم. اما روشی هم وجود دارد که بتوانیم، بقیه موارد را تغییر دهیم.

کل کد بالا را به شیوه سخت‌تری نیز می‌توانیم بنویسیم و در آن از شیء `control` استفاده نمی‌کنیم.

چون از `control` استفاده نمی‌کنیم از `matplotlib.pyplot` برای رسم نمودارها استفاده می‌کنیم.

```
import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt
```

توابع عضویت را به‌صورت مثلثی تعریف می‌کنیم (در اینجا هم می‌توانیم از روش‌های دیگر استفاده کنیم):

```
qual_lo = fuzz.trimf(x_qual, [0, 0, 5])
qual_md = fuzz.trimf(x_qual, [0, 5, 10])
qual_hi = fuzz.trimf(x_qual, [5, 10, 10])
serv_lo = fuzz.trimf(x_serv, [0, 0, 5])
serv_md = fuzz.trimf(x_serv, [0, 5, 10])
serv_hi = fuzz.trimf(x_serv, [5, 10, 10])
tip_lo = fuzz.trimf(x_tip, [0, 0, 13])
tip_md = fuzz.trimf(x_tip, [0, 13, 25])
tip_hi = fuzz.trimf(x_tip, [13, 25, 25])
```

دو ورودی 4.5 برای کیفیت غذا و 9.8 برای کیفیت سرویس را به‌صورت زیر به‌عنوان ورودی می‌دهیم:

```
qual_level_lo = fuzz.interp_membership(x_qual, qual_lo, 4.5)
qual_level_md = fuzz.interp_membership(x_qual, qual_md, 4.5)
qual_level_hi = fuzz.interp_membership(x_qual, qual_hi, 4.5)
serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, 9.8)
serv_level_md = fuzz.interp_membership(x_serv, serv_md, 9.8)
serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, 9.8)
```

حالا قوانین را به شکل زیر فعال می‌کنیم:

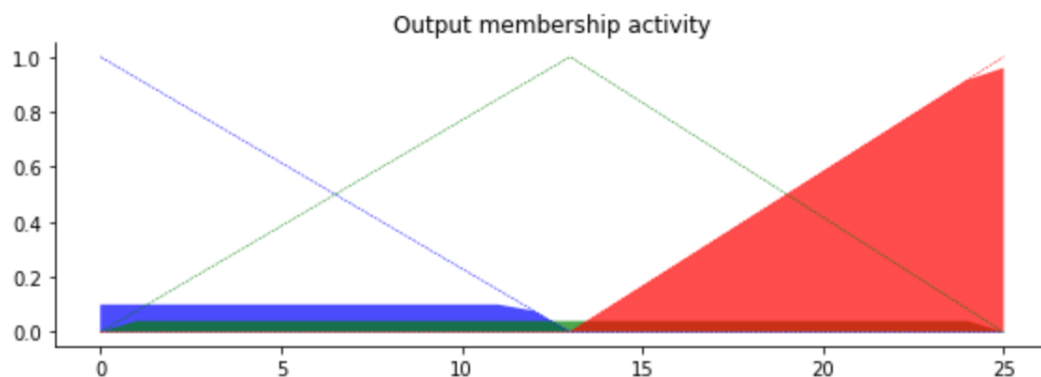
```
# Now we take our rules and apply them. Rule 1 concerns bad food OR service.
# The OR operator means we take the maximum of these two.
active_rule1 = np.fmax(qual_level_lo, serv_level_lo)
# Now we apply this by clipping the top off the corresponding output
# membership function with `np.fmin`
tip_activation_lo = np.fmin(active_rule1, tip_lo) # removed entirely to 0
# For rule 2 we connect acceptable service to medium tipping
tip_activation_md = np.fmin(serv_level_md, tip_md)
# For rule 3 we connect high service OR high food with high tipping
active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
tip_activation_hi = np.fmin(active_rule3, tip_hi)
tip0 = np.zeros_like(x_tip)
```

با دستور زیر می‌توانیم نمودار آن را رسم کنیم:

```
# Visualize these universes and membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))
ax0.plot(x_qual, qual_lo, 'b', linewidth=1.5, label='Bad')
ax0.plot(x_qual, qual_md, 'g', linewidth=1.5, label='Decent')
ax0.plot(x_qual, qual_hi, 'r', linewidth=1.5, label='Great')
ax0.set_title('Food quality')
ax0.legend()
ax1.plot(x_serv, serv_lo, 'b', linewidth=1.5, label='Poor')
ax1.plot(x_serv, serv_md, 'g', linewidth=1.5, label='Acceptable')
ax1.plot(x_serv, serv_hi, 'r', linewidth=1.5, label='Amazing')
ax1.set_title('Service quality')
ax1.legend()
ax2.plot(x_tip, tip_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_tip, tip_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_tip, tip_hi, 'r', linewidth=1.5, label='High')
ax2.set_title('Tip amount')
ax2.legend()

# Turn off top/right axes
for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
plt.tight_layout()
```

خروجی:



حالا هر سه خروجی را تجميع می کنیم:

```
aggregated = np.fmax(tip_activation_lo, np.fmax(tip_activation_md, tip_activation_hi))
```

حالا روش ديفازی را اعمال می کنیم:

```
tip = fuzz.defuzz(x_tip, aggregated, 'centroid')
tip_activation = fuzz.interp_membership(x_tip, aggregated, tip)
```

حالا خروجی را رسم و مقدار دقیق انعام را به درصد می نویسیم:

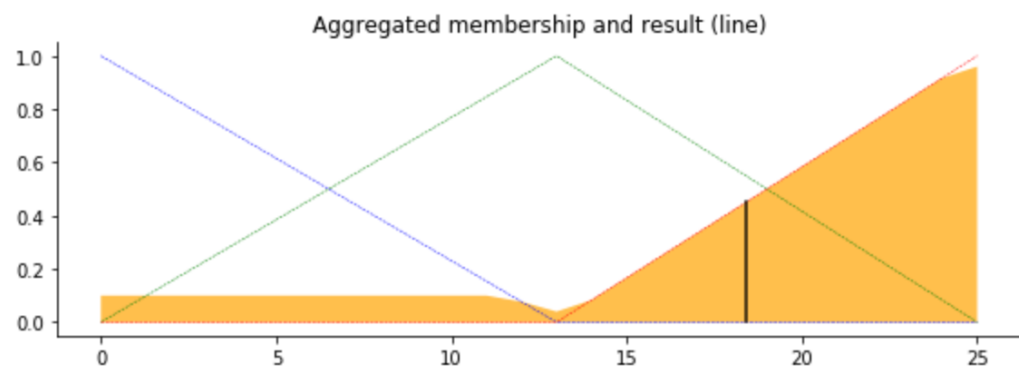
```

fig, ax0 = plt.subplots(figsize=(8, 3))
ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, aggregated, facecolor='Orange', alpha=0.7)
ax0.plot([tip, tip], [0, tip_activation], 'k', linewidth=1.5, alpha=0.9)
ax0.set_title('Aggregated membership and result (line)')
# Turn off top/right axes
for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
plt.tight_layout()
print(tip)

```

خروجی:

18.43077521848353



نتایج به نتایج کد قبلی بسیار نزدیک اما با تفاوتی جزئی است.

حالا کل کد را که در Jupyter نوشته شده است، مشاهده می‌کنیم:

Cell 1:

```

import numpy as np
import skfuzzy as fuzz
import matplotlib.pyplot as plt

# Generate universe variables
# * Quality and service on subjective ranges [0, 10]
# * Tip has a range of [0, 25] in units of percentage points

```

```

x_qual = np.arange(0, 11, 1)
x_serv = np.arange(0, 11, 1)
x_tip = np.arange(0, 26, 1)

# Generate fuzzy membership functions

qual_lo = fuzz.trimf(x_qual, [0, 0, 5])
qual_md = fuzz.trimf(x_qual, [0, 5, 10])
qual_hi = fuzz.trimf(x_qual, [5, 10, 10])
serv_lo = fuzz.trimf(x_serv, [0, 0, 5])
serv_md = fuzz.trimf(x_serv, [0, 5, 10])
serv_hi = fuzz.trimf(x_serv, [5, 10, 10])
tip_lo = fuzz.trimf(x_tip, [0, 0, 13])
tip_md = fuzz.trimf(x_tip, [0, 13, 25])
tip_hi = fuzz.trimf(x_tip, [13, 25, 25])

# Visualize these universes and membership functions
fig, (ax0, ax1, ax2) = plt.subplots(nrows=3, figsize=(8, 9))
ax0.plot(x_qual, qual_lo, 'b', linewidth=1.5, label='Bad')
ax0.plot(x_qual, qual_md, 'g', linewidth=1.5, label='Decent')
ax0.plot(x_qual, qual_hi, 'r', linewidth=1.5, label='Great')
ax0.set_title('Food quality')
ax0.legend()
ax1.plot(x_serv, serv_lo, 'b', linewidth=1.5, label='Poor')
ax1.plot(x_serv, serv_md, 'g', linewidth=1.5, label='Acceptable')
ax1.plot(x_serv, serv_hi, 'r', linewidth=1.5, label='Amazing')
ax1.set_title('Service quality')

```

```

ax1.legend()
ax2.plot(x_tip, tip_lo, 'b', linewidth=1.5, label='Low')
ax2.plot(x_tip, tip_md, 'g', linewidth=1.5, label='Medium')
ax2.plot(x_tip, tip_hi, 'r', linewidth=1.5, label='High')
ax2.set_title("Tip amount")
ax2.legend()

```

```

# Turn off top/right axes
for ax in (ax0, ax1, ax2):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
plt.tight_layout()

```

cell 2:

```

# We need the activation of our fuzzy membership functions at these values.
# The exact values 6.5 and 9.8 do not exist on our universes...
# This is what fuzz.interp_membership exists for!

qual_level_lo = fuzz.interp_membership(x_qual, qual_lo, 4.5)
qual_level_md = fuzz.interp_membership(x_qual, qual_md, 4.5)
qual_level_hi = fuzz.interp_membership(x_qual, qual_hi, 4.5)
serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, 9.8)
serv_level_md = fuzz.interp_membership(x_serv, serv_md, 9.8)
serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, 9.8)

# Now we take our rules and apply them. Rule 1 concerns bad food OR service.
# The OR operator means we take the maximum of these two.
active_rule1 = np.fmax(qual_level_lo, serv_level_lo)

# Now we apply this by clipping the top off the corresponding output

```

```

# membership function with `np.fmin`
tip_activation_lo = np.fmin(active_rule1, tip_lo) # removed entirely to 0
# For rule 2 we connect acceptable service to medium tipping
tip_activation_md = np.fmin(serv_level_md, tip_md)
# For rule 3 we connect high service OR high food with high tipping
active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
tip_activation_hi = np.fmin(active_rule3, tip_hi)
tip0 = np.zeros_like(x_tip)
# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))
ax0.fill_between(x_tip, tip0, tip_activation_lo, facecolor='b', alpha=0.7)
ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.fill_between(x_tip, tip0, tip_activation_md, facecolor='g', alpha=0.7)
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, tip_activation_hi, facecolor='r', alpha=0.7)
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.set_title('Output membership activity')
# Turn off top/right axes
for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
plt.tight_layout()

```

cell 3:

```

# Aggregate all three output membership functions together
aggregated = np.fmax(tip_activation_lo, np.fmax(tip_activation_md, tip_activation_hi))
# Calculate defuzzified result
tip = fuzz.defuzz(x_tip, aggregated, 'centroid')

```

```

tip_activation = fuzz.interp_membership(x_tip, aggregated, tip) # for plot
# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))
ax0.plot(x_tip, tip_lo, 'b', linewidth=0.5, linestyle='--', )
ax0.plot(x_tip, tip_md, 'g', linewidth=0.5, linestyle='--')
ax0.plot(x_tip, tip_hi, 'r', linewidth=0.5, linestyle='--')
ax0.fill_between(x_tip, tip0, aggregated, facecolor='Orange', alpha=0.7)
ax0.plot([tip, tip], [0, tip_activation], 'k', linewidth=1.5, alpha=0.9)
ax0.set_title('Aggregated membership and result (line)')
# Turn off top/right axes
for ax in (ax0,):
    ax.spines['top'].set_visible(False)
    ax.spines['right'].set_visible(False)
    ax.get_xaxis().tick_bottom()
    ax.get_yaxis().tick_left()
plt.tight_layout()
print(tip)

```


همانطور که در کدها مشاهده می‌کنید، کدهای cell2 و cell3 اکثراً از numpy استفاده کرده‌اند و نه خود Skfuzzy. و این قسمت‌ها جایی است که شما می‌توانید در صورت علاقه کد خود را بر اساس روش‌های inference دیگر تغییر دهید و یا روش T-norm و S-norm خود را تغییر دهید.

یعنی قسمت‌هایی که در زیر مشخص شده‌اند:

```

serv_level_lo = fuzz.interp_membership(x_serv, serv_lo, 9.8)
serv_level_md = fuzz.interp_membership(x_serv, serv_md, 9.8)
serv_level_hi = fuzz.interp_membership(x_serv, serv_hi, 9.8)
# Now we take our rules and apply them. Rule 1 concerns bad food OR serv
# The OR operator means we take the maximum of these two.
active_rule1 = np.fmax(qual_level_lo, serv_level_lo)
# Now we apply this by clipping the top off the corresponding output
# membership function with `np.fmin`
tip_activation_lo = np.fmin(active_rule1, tip_lo) # removed entirely to
# For rule 2 we connect acceptable service to medium tipping
tip_activation_md = np.fmin(serv_level_md, tip_md)
# For rule 3 we connect high service OR high food with high tipping
active_rule3 = np.fmax(qual_level_hi, serv_level_hi)
tip_activation_hi = np.fmin(active_rule3, tip_hi)
tip = np.zeros_like(x_tip)
# Visualize this
fig, ax0 = plt.subplots(figsize=(8, 3))
ax0.fill_between(x_tip, tip0, tip_activation_lo, facecolor='h', alpha=0

```

```
38]:  # Aggregate all three output membership functions together  
    aggregated = np.fmax(tip_activation_lo, np.fmax(tip_activation_md, tip_activation_hi))  
    # Calculate defuzzified result  
    tip = fuzz.defuzz(x_tip, aggregated, 'centroid')
```

البته کتابخانه‌هایی در زبان‌های دیگر وجود دارند که متدهای بیشتری برای T-norm، S-norm و inference method در اختیار شما قرار می‌دهند.

منبع اصلی:

<https://pythonhosted.org/scikit-fuzzy/>