experta

Expert systems in Python

- Install Python

- Install experta :
  - cmd:

    ```
    pip install experta
    ```

  - Python script (within python code) :

    ```
    import sys
    import subprocess
    subprocess.check_call([sys.executable, '-m', 'pip', 'install', 'experta'])
    ```

- **Facts Manipulation**

- **Fact is a subclasse of dict**
  - **Create facts :**

```
from experta import *
faits = Fact('one', 'two', 'x', 'z', a=5, b=10, color='green')
print(faits[1])
print(faits["a"])
print(faits["color"])
```

Output :

two
5
green

- **Rules Manipulation**

- **A rule is created using the decorator @Rule**
- **A rule is composed of two parts: Premise and Action**
  - **Create rules:**

```
from experta import *
class Light(Fact):
        pass
@Rule(Light(color="red"))  #Prémisse
def Red_Light(self):          #Action
    #Action to do
```

- **Inference engine (KnowledgeEngine)**
  - **Ceate a subclasse for the engine and use Rule in order to activate the methods:**

```
from experta import *
class CrossStreet(KnowledgeEngine):
 @DefFacts()                              #Dectare initial facts (Fact base)
    def Initial_Facts(self):
     #initial facts
 @Rule(Light(color="red"))  #Premise
    def Red_Light(self):        #Action
      #Action to do
engine = CrossStreet()                    #instanciate the engine class
engine.reset()                            #Use of Initial facts (DefFacts())
engine.declare(Light(color="red"))    #Declare new facts
engine.run()                              #Run the engine cycle
```

- **Example of Expert system (Lights) / if_statement**

```
from random import choice

color = choice(['green','red','yellow','blinking-yellow'])
if color=='green':
 print("Walk")
elif color=='red':
 print("Don't walk")
elif color=='yellow':
 print("Be cautious !")
else:
 print("Be cautious !!!\n Be or not to be")
```

- **Example of Expert system (Lights) / experta**

```
from experta import *


class Light(Fact):
     pass

class RobotCrossStreet(KnowledgeEngine):

    @DefFacts()
    def initial_facts(self):
        yield Light(color="white")

    @Rule(Light(color='green'))
    def green_light(self):
        print("Walk")

    @Rule(Light(color='red'))
    def red_light(self):
        print("Don't walk")

    @Rule(Light(color='yellow'))
    def cautious(self):
        print("Be cautious !!!")
engine = RobotCrossStreet()
engine.reset()
colorselect = input("Enter a color :")
print("Color is :",colorselect)
engine.declare(Light(color=colorselect))
engine.run()
```

Output :

```
Enter a color :red
Color is : red
Don't walk
```

see file: Light.py

- **Use of logical operators**

```
    @Rule(OR(Light(color='yellow'),Light(color='blinking-yellow')))
    def cautious(self):
        print("Be cautious !!!")

engine = RobotCrossStreet()
engine.reset()
colorselect = choice(['green', 'yellow', 'blinking-yellow', 'red'])
print("Color is :",colorselect)
engine.declare(Light(color=colorselect))
engine.run()
```

Output

```
Color is : yellow
Be cautious !!!
```

```
Color is : blinking-yellow
Be cautious !!!
```

- **Preview the list of facts (engine.facts)**

```
    @DefFacts()
    def initial_facts(self):
        yield Light(color="white")

    @Rule(Light(color='green'))
    def green_light(self):
        print("Walk")

    @Rule(Light(color='red'))
    def red_light(self):
        print("Don't walk")

    @Rule(OR(Light(color='yellow'),Light(color='blinking-yellow')))
    def cautious(self):
        print("Be cautious !!!")

engine = RobotCrossStreet()
engine.reset()
colorselect = choice(['green', 'yellow', 'blinking-yellow', 'red'])
print("Color is :",colorselect)
engine.declare(Light(color=colorselect))
engine.run()
print(engine.facts)
```

Output

```
Color is : red
Don't walk
<f-0>: InitialFact()
<f-1>: Light(color='white')
<f-2>: Light(color='red')
```