Step 2 my approach to this question would be focus on some cases in the grammar rule, it will have NP-> noun directly rather than in the normal CNF form, which has two components on the right hand side.In the getgrammar synatx rules function, I want this function to accept rule in 3 length form, and a the last element =-10 help to determine the which rule should use in the CYK parse function. In the CYK function, if the last element is -10 then it means from a C1F form rule, otherwise from the normal CNF form rule. Because C1F rule comes directly from 1 component, so I test it from the initial word, and for every word in position k, once it qualifies, put it into the upper class with a rightchild None(for example NP->noun, so noun will directly go to NP class). In the walktree function, I ingnore to track on the right child if it is already None.

In [ ]:
```
    for i, j, k in subspans(len(words)):
        for X, Y, Z, p in getGrammarSyntaxRules(grammar):
            #print(1)
            printV('i:', i, 'j:', j, 'k:', k, '', X, '->', Y, Z, '['+str(p)+']',
                   'PYZ =' ,getP(Y, i, j), getP(Z, j+1, k), p, '=', getP(Y, i, j) * g

            if p!=-10:

                PYZ = getP(Y, i, j) * getP(Z, j+1, k) * p
                if PYZ > getP(X, i, k):
                    print("daole daole")
                    printV('     inserting from', i, '-', k, ' ', X, '->', T[Y+'/'+str
                           'because', PYZ, '=', getP(Y, i, j), '*', getP(Z, j+1,
                           'getP(' + X + ',' + str(i) + ',' + str(k) + ')')
                    P[X + '/' + str(i) + '/' + str(k)] = PYZ

                    T[X + '/' + str(i) + '/' + str(k)] = Tree.Tree(X, T[Y+'/'+str(i)+
            elif p==-10:
                global flag
                if getP(Y, 0, 0)!=0 and flag==0:

                    PYZ = getP(Y, 0, 0)
                    if PYZ > getP(X, 0, 0):
                        flag += 1
                        P[X + '/' + str(0) + '/' + str(0)] = PYZ
                        T[X + '/' + str(0) + '/' + str(0)] = Tree.Tree(X, T[Y + '/' +
                PYZ = getP(Y, k, k)
                if PYZ > getP(X, k, k):

                    P[X + '/' + str(k) + '/' + str(k)] = PYZ
                    T[X + '/' + str(k) + '/' + str(k)] = Tree.Tree(X, T[Y + '/' + str


def getGrammarSyntaxRules(grammar):


    for rule in grammar['syntax']:

        if len(rule) ==4:
            yield rule[0], rule[1], rule[2], rule[3]

        if len (rule) ==3:
            yield rule[0], rule[1], rule[2],-10




    def walkTree(self, l):
        if (self.leftChild is None):
            l.append([self.categoryName, self.lexiconItem])
        elif (self.rightChild is None):
```

```
            self.leftChild.walkTree(1)
        else:
            self.leftChild.walkTree(1)
            self.rightChild.walkTree(1)
```

Step 3

if I have CYKParse(['what','is','temperature','in', 'Irvine', 'yesterday'], CYKParse.getGrammarWeather()), in this example I donot have the proposition the and make temperature a single word, and it will belongs NP and connecting the sentence .So I change the grammarweather with ['NP', 'Noun', 0.7],then to the sentence it should recongnize a single word noun "temperature" as NP and do the rest combination.

In [ ]:
```
['NP', 'Noun', 0.7],
['NP', 'Article', 'Noun', 0.1],
['NP', 'Adjective', 'Noun', 0.2],
```

Step 4

True, fig 23.3 shows each word has fixed probability to occur, and in the syntax rule, two words will combine together first, in the CYK algorithm is combining word Xi,j Xj,k into Xi,k, as a result grammar rules in the CNF will be linked together first, and C1F form will not impact.but if we have two situations for the sentences "what is temperature in Tustin yesterday" "what is the temperature in Tustin yesterday" In the CNF [NP+AdverbPhrase[NP[Article the][Noun temperature]] in the C1F [NP+AdverbPhrase[Noun temperature[AdverbPhrase] because article will disappear and donot invovle in the formation of parse tree.

In [ ]:
```
T, P = CYKParse.CYKParse(['what', 'is', 'the', 'temperature', 'in', 'Tustin', 'yesterd
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

    #S/0/6 [S[WQuestion what][VP[Verb is][NP+AdverbPhrase[NP[Article the][Noun tempera
T, P = CYKParse.CYKParse(['what', 'is',  'temperature', 'in', 'Tustin', 'yesterday'],
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

    # S/0/5 [S[WQuestion what][VP[Verb is][NP+AdverbPhrase[Noun temperature][AdverbPhr
```

Step 5 part1

I use the get temperature to update the data from different cities, and as the ouput shows the weather in yesterday is 101. I add different get temperature statements for different time, and for getGrammarWeather, I add places and adverb of different time

In [ ]:
```
def getTemperature(location, time):
    if location == 'Pasadena':
        if time == 'now':
            return '90'
        elif time == 'tomorrow':
            return '45'
        elif time == 'yesterday':
            return '111'
    elif location == 'Tustin':
        if time == 'now':
            return '100'
```

```
        elif time == 'tomorrow':
            return '10'
        elif time == 'yesterday':
            return '101'

    elif location == 'Irvine':
        if time == 'now':
            return '68'
        elif time == 'tomorrow':
            return '70'
        elif time == 'yesterday':
            return '105'
    else:
        return 'unknown'

    #revised in getGrammarWeather part

            ['Name', 'Peter', 0.1],
            ['Name', 'Sue', 0.1],
            ['Name', 'Irvine', 0.3],
            ['Name', 'Pasadena', 0.3],
            ['Name', 'Tustin', 0.2],
            ['Pronoun', 'I', 1.0],
            ['Adverb', 'now', 0.25],
            ['Adverb', 'today', 0.25],
            ['Adverb', 'tomorrow', 0.25],
            ['Adverb', 'yesterday', 0.25],
            ['Preposition', 'with', 0.5],
            ['Preposition', 'in', 0.5],



T, P = CYKParse.CYKParse(['what', 'is', 'the', 'temperature', 'in', 'Tustin', 'yesterd
    sentenceTree = getSentenceParse(T)
    updateRequestInfo(sentenceTree)
    reply()

# Peter, the temperature in Tustin yesterday is 101.
```

Step 5 part2

This part, I use the sentence['will','yesterday','be','hotter', 'than','tomorrow','in','Pasadena'] and the reply is "Peter, yesterday's temperature is hotter than tomorrow" I use the leaf[1] to detect whether the questions comes from will, then I record the time1, time2 which contains two time spot. And then compare it to print the result.

In [ ]:

```
def updateRequestInfo(Tr):
    global comparison
    global requestInfo
    lookingForLocation = False
    lookingForName = False
    for leaf in Tr.getLeaves():
        print(leaf)
        if leaf[1]=="will":
            comparison=True
    if leaf[0] == 'Adverb':
            print(leaf[0],leaf[1])
            if requestInfo['time']!='' and comparison:
                requestInfo['time2'] = leaf[1]
            else:
                requestInfo['time'] = leaf[1]
# getGrammarWeather part
```

```
            ['WQuestion', 'will', 0.25],
            ['Adjective', 'hotter', 0.5],
            ['Adjective', 'my', 0.5],
            ['conjun', 'than', 0.25],

def reply():
    if comparison:
        a= getTemperature(requestInfo['location'], requestInfo['time'])
        b= getTemperature(requestInfo['location'],requestInfo['time2'])
        print(a,requestInfo['time'])
        print(b, requestInfo['time2'])

        if (int(a)>int(b)):

            print(salutation + requestInfo['time'] +"'s temperature is hotter than " +
        elif (int(a)==int(b)):
            print(salutation + " , these two days are same temperature")
        elif a=="unkown":
            print("sorry cannot compare unknown")
        else:
            print(salutation + requestInfo['time2'] + "'s temperature is hotter than

    else:
        print(salutation + 'the temperature in ' + requestInfo['location'] + ' ' +
            time + ' is ' + getTemperature(requestInfo['location'], time) + '.')
```