In [1]:
```
# This code is a QA system that answers questions starting with Who, What, Where, or When.
# It uses various functions and regex to extractr information from the input question
#and find the corresponding answer
```

In [2]:
```
!pip install wikipedia
```

```
Requirement already satisfied: wikipedia in c:\users\marya\anaconda3\lib\site-packages (1.4.0)
Requirement already satisfied: beautifulsoup4 in c:\users\marya\anaconda3\lib\site-packages (from wi
kipedia) (4.12.2)
Requirement already satisfied: requests<3.0.0,>=2.0.0 in c:\users\marya\anaconda3\lib\site-packages
(from wikipedia) (2.31.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\marya\anaconda3\lib\site-package
s (from requests<3.0.0,>=2.0.0->wikipedia) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\marya\anaconda3\lib\site-packages (from requ
ests<3.0.0,>=2.0.0->wikipedia) (3.4)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\marya\anaconda3\lib\site-packages (fro
m requests<3.0.0,>=2.0.0->wikipedia) (1.26.16)
Requirement already satisfied: certifi>=2017.4.17 in c:\users\marya\anaconda3\lib\site-packages (fro
m requests<3.0.0,>=2.0.0->wikipedia) (2024.7.4)
Requirement already satisfied: soupsieve>1.2 in c:\users\marya\anaconda3\lib\site-packages (from bea
utifulsoup4->wikipedia) (2.4)
```

In [*]:
```
#This will import the necessary libraries
import wikipedia
import sys
import os
import lxml
from nltk.chunk import tree2conlltags
import re
from datetime import datetime

#import specific functions from nltk libraries
from nltk import ne_chunk, pos_tag, word_tokenize
```

In [*]:
```
#Downloading necessary NLTK datasets
import nltk

nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('maxent_ne_chunker')
nltk.download('words')
```

In [*]:
```
 # Define a tokenizer function that applies named entity recognition (NER) on a given sentence
def tokenizer(sent):
    words = nltk.word_tokenize(sent) # tokenize sentence into words
    pos_tags = nltk.pos_tag(words) # tag words with part of speech
    tree = nltk.ne_chunk(pos_tags) # identifies named entities in the sentence
    conll_tags = nltk.tree2conlltags(tree) #convert named entities to CONLL format
    return conll_tags

```

In [*]:
```
# an example of using the tokenizer function
tokenizer('This is Northern Virginia')
```

In [ ]:
```
%%markdown
[\d]+: one or more digits (0-9)
\s: a whitespace character (space, tab, newline, etc.)
(\w+): one or more word characters (letters, digits, or underscore) - corrected to properly capture a
\s: another whitespace character
[\d]+: one or more digits again
```

In [ ]:
```
get_question("When was George Washington born?".title())
```

In [14]:
```
"""
The function get_question takes a string s as unput and returns a matching question
based on the provided rules. The regular expression in the rules check for certain patterns
in the string and return the corresponding groups
"""
def get_question(s):
    rules = [[r'Where (Is|Was) (.+)', ["{0}"]],
             [r'Who (Is|Was) (.+)', ["{0}"]],
             [r'What (Is|Was) (.+) Age', ["{0}"]],
             [r'What (Is|Was) (.+)', ["{0}"]],
             [r'When (Is|Was) (.+) Born', ["{0}"]],
             [r'When (Is|Was) (.+) Birthday', ["{0}"]],
             [r'When (Is|Was) (.+)', ["{0}"]],
            ]
    for r, c in rules:
        m = re.match(r, s.rstrip('.!?'))
        if m:
            return m.groups()[1]
    return ""
```

In [15]:
```
get_question("Where is the capital city of virginia?".title())
```

Out[15]:
```
'The Capital City Of Virginia'
```

In [16]:
```
"""
The function get_ans_wiki takes a query string and attempts to fetch a summary from Wikipedia
using the wikipedia.summary() function. It handles exceptions if the summary is not found.
"""
def get_ans_wiki(query):
    try:
        return wikipedia.summary(query, sentences=1)
    except wikipedia.exceptions.DisambiguationError as e:
        return wikipedia.summary(e.options[1], sentences=1)
    except wikipedia.exceptions.PageError:
        return ""
    except Exception:
        return ""
```

In [17]:
```
get_ans_wiki('The Capital City of Virginia')
```

Out[17]:
```
'Capital One Tower is a high-rise office building in Capital One Center, a mixed-use development adj
acent to the McLean station in Tysons, Virginia.'
```

In [18]:
```
"""
The function check_digits takes a text input and checks for specific patterns using
regular expressions. It returns the matching groups if a pattern is found.
"""
def check_digits(text):
    return re.findall(r'\d+', text)
```

In [19]:
```
get_question('When is Independence day of America?'.title())
```

Out[19]:
```
'Independence Day Of America'
```

In [20]:
```
tokenizer('When was Taylor Swift born?'.title())
```

Out[20]:
```
[('When', 'WRB', 'O'),
 ('Was', 'NNP', 'B-PERSON'),
 ('Taylor', 'NNP', 'I-PERSON'),
 ('Swift', 'NNP', 'I-PERSON'),
 ('Born', 'NNP', 'I-PERSON'),
 ('?', '.', 'O')]
```

```
In [21]: get_ans_wiki(get_question('When was Taylor Alison Swift born?'.title()))

Out[21]: 'Taylor Alison Swift (born December 13, 1989) is an American singer-songwriter.'
```

```
In [22]: check_digits(get_ans_wiki(get_question('When was Taylor Alison Swift born?'.title())))

Out[22]: ['13', '1989']
```

```
In [23]: """
         The transform() function takes two arguments: 'u' and 'q'
         It removes any punctuation at the end of 'u' and then applies different regular expressions to
         transform the question. If any of the transformation is successful, it will return the transformed que
         with 'q' concatenated and a period added at the end.
         """
         def transform(u, q):
             u = u.rstrip('.!?')
             ur = re.sub(r'When (Is|Was) (.+) Born', r'\2 Was Born On ', u)
             if ur != u:
                 return ur + str(q) + '.'

             ur = re.sub(r'When (Is|Was) (.+) Birthday', r'\2 Birthday Is On ', u)
             if ur != u:
                 ur = ur + q
                 ur = re.sub(r'\d{4}', '', ur)
                 return ur + '.'

             ur = re.sub(r'What (Is|Was) (.+) Age', r'\2 Age Is ', u)
             if ur != u:
                 return ur + str(q) + '.'
```

```
In [24]: transform("when is Taylor Swift Born?".title(), "1989")

Out[24]: 'Taylor Swift Was Born On 1989.'
```

```
In [25]: transform("When was George Washington Born?".title(), 1732)

Out[25]: 'George Washington Was Born On 1732.'
```

```
In [ ]: """
        The main() function is the main entry point of the program.
        It initializes a log file, prompts the user for input, and processes the input to find the answer.
        It uses the tokenizer() and get_question() functions to prepare the input for further processing.
        Based on the type of question ('Where', 'What', 'Who', 'When', or 'Age'), it retrives
        information from Wikipedia using the get_ans_wiki() function. For 'Who' and 'Age' questions,
        it also uses the check_digits() function to extract relevant information.
        Finally, it prints the answer to the user and writes it to the log files.
        """

        def main():
            try:
                log = open('mylogfile.txt', "w+")
                print("*** This is a QA system by Maryah Austria. It will try to answer questions that start w
                counter = 1
                while True:
                    s = input()
                    e = s

                    if s == "exit":
                        print("\nThank you! Goodbye.")
                        break

                    log.write(str(counter) + "Ques) " + s + "\n")
                    flag = False
                    final = ""
                    wiki = ""

                    if s != "" and s is not None:

                        s = s.title()
                        n = tokenizer(s)
                        r = get_question(s)

                        if r!= "":
                            if ("Where" in s or "What" in s):

                                wiki = get_ans_wiki(r)
                                match = re.search(r'\d{4}', wiki)

                                if match is not None:
                                    wiki = ""

                            if ('I-PERSON' or 'B-PERSON') in n[2]:
                                flag = True

                            if "Who" in s and flag:
                                wiki = get_ans_wiki(r)

                            if ("When" in s) and flag:

                                wiki = get_ans_wiki(r)
                                result = check_digits(wiki)

                                if result != "":
                                    for dd in result:
                                        if(len(dd)==4):
                                            match = dd
                                            break

                                    appends = transform(s, match)
                                    if appends is not None:
                                        final = appends

                                else:
                                    final = ""

                            else:
                                final = get_ans_wiki(r)
                        else:
                            final = ""
            else:
                print("Please ask a valid question!")

            if final == "":
                log.write(str(counter) + "A) Answer not found.\n\n")
                print("I am sorry, I don't know the answer.\n")

            else:
                try:
                    log.write(str(counter) + "A) " + final + "\n\n")
                    print(final + "\n")
                except Exception as GeneralException:
                    log.write(str(counter) + "A) Answer not found.\n\n")
                    print("I am sorry, I don't know the answer.\n")

            counter = counter + 1
            print("Please ask another question or type 'exit' to exit from the program:\n")
    except Exception as GeneralException:
        print(GeneralException)
    finally:
        log.close()

main()
```

```
*** This is a QA system by Maryah Austria. It will try to answer questions that start with Who, Wha
t, Where, or When. Enter 'exit' to leave the program.
When was George Washington born?
George Washington Was Born On 1732.

Please ask another question or type 'exit' to exit from the program:
```

In [ ]:
```
When was George Washington born?
```

In [ ]: