In [1]:
```
# Install necessary packages
!pip install textstat
```

```
Requirement already satisfied: textstat in c:\users\alon\anaconda3\lib\site-p
ackages (0.5.4)
Requirement already satisfied: pyphen in c:\users\alon\anaconda3\lib\site-pac
kages (from textstat) (0.9.5)
Requirement already satisfied: repoze.lru in c:\users\alon\anaconda3\lib\site
-packages (from textstat) (0.7)
```

In [2]:
```
simple_text = (
"You can go to a restaurant to eat breakfast, lunch, or "
"dinner and even a snack when you want. Restaurants serve "
"food to you and to other people who go and eat at them. "
"First, you look at a menu that lists all of the foods and drinks "
"that you can order from the restaurant. "
)

complex_text = (
"Playing games has always been thought to be important to "
"the development of well-balanced and creative children; "
"however, what part, if any, they should play in the lives "
"of adults has never been researched that deeply. I believe "
"that playing games is every bit as important for adults "
"as for children. Not only is taking time out to play games "
"with our children and other adults valuable to building "
"interpersonal relationships but is also a wonderful way "
"to release built up tension."
)
```

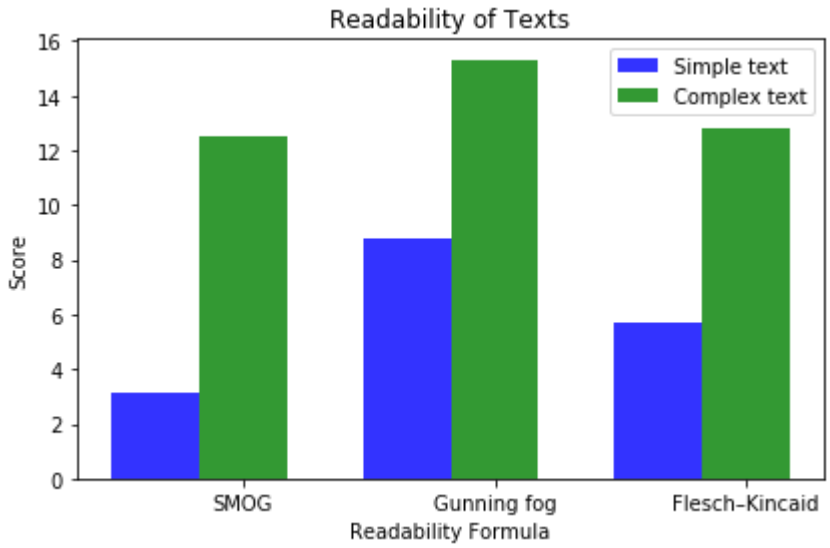
Readability

Readability is a measure of how easy it is to read and understand a written text. Lower readability implies that less reading comprehensionn is required, and higher readability implies that more reading comprehensionn is required.

In [3]:
```
import textstat
import numpy as np
import matplotlib.pyplot as plt

# In: Text to measure readability
# Out: Array of readability measurements (smog_index, gunning_fog, flesch_kinc
aid_grade)
def calcReadability(text):
    v1 = textstat.smog_index(text)
    v2 = textstat.gunning_fog(text)
    v3 = textstat.flesch_kincaid_grade(text)

    return [v1, v2, v3]

def plotReadability(text1, text2, label1='Text 1', label2='Text 2'):
    # data to plot
    n_groups = 3
    readability_1 = calcReadability(text1)
    readability_2 = calcReadability(text2)

    print(readability_1)
    print(readability_2)

    # create plot
    fig, ax = plt.subplots()
    index = np.arange(n_groups)
    bar_width = 0.35
    opacity = 0.8

    rects1 = plt.bar(index, readability_1, bar_width,
    alpha=opacity,
    color='b',
    label=label1)

    rects2 = plt.bar(index + bar_width, readability_2, bar_width,
    alpha=opacity,
    color='g',
    label=label2)

    plt.xlabel('Readability Formula')
    plt.ylabel('Score')
    plt.title('Readability of Texts')
    plt.xticks(index + bar_width, ('SMOG', 'Gunning fog', 'Flesch-Kincaid'))
    plt.legend()

    plt.tight_layout()
    plt.show()
```

In [4]:
```
plotReadability(simple_text, complex_text, 'Simple text', 'Complex text')
```

```
[3.1, 8.77, 5.7]
[12.5, 15.31, 12.8]
```



Type-Token Ratio (TTR)

Type-Token ratio (TTR) is a measure of lexical complexity. The higher the TTR, the larger the amount of lexical variation of a written text.

In [5]:
```
def calcTTR(text):
    # Number of words
    token_count = len(text.split());

    counts = dict()
    words = text.split()

    for word in words:
        if word in counts:
            counts[word] += 1
        else:
            counts[word] = 1

    # Number of unique words
    type_count = len(counts);

    return type_count / token_count
```

In [6]:
```
print('TTR:')
print('Simple text: ' + str(calcTTR(simple_text)))
print('Complex text: ' + str(calcTTR(complex_text)))
```

```
TTR:
Simple text: 0.6909090909090909
Complex text: 0.7560975609756098
```

Syntax Tree Depth

The depth of a sentence's syntax tree is a measure of the sentence's complexity.

In [7]:
```
from nltk.parse.corenlp import *
from nltk.tokenize import sent_tokenize, word_tokenize

def calcDepth(text):
    parser = CoreNLPParser()

    def calcSingleDepth(sent):
        parse = next(parser.raw_parse(sentence))
        #parse.pretty_print()
        return parse.height()

    sentences = sent_tokenize(text)
    totalDepth = 0

    for i in range(len(sentences)):
        sentence = sentences[i]
        totalDepth += calcSingleDepth(sentence)

    return totalDepth / len(sentences)
```

In [8]:
```
print('Sentence syntax tree depth:')
print('Simple text: ' + str(calcDepth(simple_text)))
print('Complex text: ' + str(calcDepth(complex_text)))
```

```
Sentence syntax tree depth:
Simple text: 14.333333333333334
Complex text: 18.666666666666668
```

Resources

- Readability Python Package: https://pypi.org/project/textstat/ (https://pypi.org/project/textstat/)
- Stanford CoreNLP: https://stanfordnlp.github.io/CoreNLP/download.html (https://stanfordnlp.github.io/CoreNLP/download.html)
- CoreNLP Server Guide: https://stanfordnlp.github.io/CoreNLP/corenlp-server.html#getting-started (https://stanfordnlp.github.io/CoreNLP/corenlp-server.html#getting-started)