# CSCA08H Worksheet: Random Story Generation

## 1 Introduction

You are going to develop a program that can read a file containing a story and then write a new story based on a random selection of the words from the original story.

As an example, consider this old German saying:

```
What I spent, I had; what I saved, I lost; what I gave, I have.
```

We are going to write a new saying using only those words, in somewhat random order. Here are the rules for choosing a random word:

- Start with a single word, such as `I`, and print the word. We will call that word the *current context.*
- Look for all occurrences of the current context and pick one of the words that follow the current context. In our example, here are the words that follow `I`:

  ```
  spent,  had;  saved,  lost;  gave,  have.
  ```

  Punctuation is included as part of the word, and capitalisation matters. For example, `what` and `What` are considered to be different words.

  After you pick the next word, print the word, with a space between it and the previous word. The word that you picked becomes the next current context, and we repeat the process. For example, if we pick `lost;`, there is only one word that follows it: `what`. We pick `what` as our next word, print a space and `what`, and then make `what` the next current context. So far, we have printed this:

  ```
  I lost; what
  ```

- Keep repeating the process until you decide that you have printed enough words. Note that a word may be printed more than once.

Task 1 Assume that the current context is `what`. Write the words that follow the current context (if the same word appears more than once, write it down again):

I I

Task 2 In the previous example, the current context was only one word long, but you can imagine using 2 words for the current context. For example, if the current context was `what I` then here are the words that follow this context:

```
saved,  gave,
```

Let's say we pick `gave,`. We print `gave,`, and then the new 2-word current context is `I gave,`.

Assume that the current context is `I gave,`. Write the word(s) that follow the current context:

I

## 2 A Larger Example

Your program will get the following information:

- **Training text**: the text that the new story will be based on, such as the German saying above, or a poem.
- **Number of words in context**: the number of words of context from the training text to use to determine the next word.
- **Length of story**: the number of words to print in the story.

Here is some training text. Treat newlines like spaces.

```
The sun did not shine.
It was too wet to play.
So we sat in the house
All that cold, cold, wet day.
I sat there with Sally,
We sat there we two.
And I said, "How I wish
We had something to do!"
Too wet to go out
And too cold to play ball.
So we sat in the house.
We did nothing at all.
So all we could do was to
Sit! Sit! Sit! Sit!
And we did not like it.
Not one little bit.
```

Using a 1-word context:

1. Assume that the current context is `to`. Write the words that follow the current context:

   play. do!" go play Sit!

2. Assume that the current context is `wet`. Write the words that follow the current context:

   to day. to

Using a 2-word context:

1. Assume that the current context is `did not`. Write the words that follow the current context:

   shine. like

2. Choose one of those words randomly. Now print the new current context:

   not shine.

If the length of the story to be printed is **11 words** and there is **one word of context**, here are some possible stories:

- cold, wet to play ball. So we two. And too cold
- could do was too wet to play. So we sat in
- said, "How I sat there with Sally, We did nothing at
- I said, "How I sat in the house. We had something
- all we could do was to go out And too cold
- Sit! Sit! Sit! Sit! Sit! Sit! Sit! Sit! Sit! Sit! Sit!

Finish writing the following story (11 words long, using one word of context):

We did not shine. It was ____________________

## 3 Representing the Data

We need to decide how to represent the data for the random story generation problem.

- **The new story**:
  Which Python type should we use to represent this data? str List[str]
- **The number of words in the new story**:
  Which Python type should we use to represent this data? int
- **The number of words of context**:
  Which Python type should we use to represent this data? int
- **The context and next word(s)**:
  To determine how to represent this information, let's start by writing down what we need to keep track of. We'll use this sample training text:

  ```
  So, as fast as I could, I went after my net.
  ```

  Using **one word** of context, complete the table below:

| Context (one word) | Possible next words |
|---|---|
| So, | as |
| as | fast I |
| fast | as |
| I | could, went |
| could, | I |
| went | after |
| after | my |
| my | net. |
| net. | ? |

What to do with 'net.' / 'my net.' ?
- store empty list as the value
- not store 'net.' as a key at all
- start at the beginnig:
  store 'So,' as the value
- store a randomly selected word
  as the value

  Using **two words** of context, complete the table below:

| Context (two words) | Possible next words |
|---|---|
| So, as | fast |
| as fast | as |
| fast as | I |
| as I | could, |
| I could, | I |
| could, I | went |
| I went | after |
| went after | my |
| after my | net. |
| my net. | ? |

  Which Python type should we use to represent this data? Write it as you would in a type contract.

  dictionary Dict[tuple, List[str]] Dict[str, List[str]]

$$\text{Context} \xrightarrow{\qquad\qquad} \text{Possible next words}$$

but keys must be immutable!

- str
- ~~List[str]~~
- tuple of strs

List[str]

## 4 Working with the Data

We will represent the words in the original story as a list of strings. You'll need to read the words in the file and create that list. This string method will help:

```
>>> help(str.split)
Help on method_descriptor:

split(self, /, sep=None, maxsplit=-1)
    Return a list of the words in the string, using sep as the delimiter string.

    sep
      The delimiter according which to split the string.
      None (the default value) means split according to any whitespace,
      and discard empty strings from the result.
    maxsplit
      Maximum number of splits to do.
      -1 (the default value) means no limit.
```

If a variable `training_file` is a file open for reading, then `training_file.read()` returns a string containing the entire contents of the file. You can use method `split` to make your list.

Write some Python statements to do this, assigning the list to variable `story`:

```
training_file = open('original_story.txt', 'r')
```

content = training_file.read()
story = content.split()

Here is an example of a story list:

```
  0      1      2        3      4      5        6      7      8         9      10     11          length: 12
['And', 'the', 'fan,', 'and', 'the', 'cup,', 'And', 'the', 'ship,', 'and', 'the', 'fish.']
```

start

stop

We will represent the context information as a dictionary where the keys are tuples of strings and the values are lists of strings. For the story above, provide the dictionary with a context length of 2. We have given you the first key below; write the list of values for it.

Fill in the dictionary with all the 2-word contexts and their values. Reminder: Python is case sensitive, so `'And'` and `'and'` are not equal.

`{ ('And', 'the') : [` 'fan,', 'ship,'],

('the', 'fan,'): ['and'],
('fan,', 'and'): ['the'],
('and', 'the'): ['cup,', 'fish.']
('the', 'cup,'): ['And'],
('And', 'the'): ['ship,'],
('the', 'ship,'): ['and'],
('ship,', 'and'): ['the']

`}`

## 5 Planning the Program

Now that you have determined how to represent the data, add the type contract to the function header below. Next, plan your program. Start by listing the tasks as comments, then replace them with function calls and write the corresponding function headers and docstrings.

```
def generate_random_story(training_file:TextIO, context_length:int , num_words:int ) ->     :
    """Return a randomly generated story with num_words words based on a context
    of context_length words from the text in training_file.
    """
```

Three helper functions:

1. read the input file, make a list of words
2. make the context to possible next words dictionary from the list
3. generate the story using the dictionary