Introduction to Computer Science I
MCS 177 - Spring 2020
Project 4: Cryptanalysis
Due 3/20

Overview

This project is designed to reinforce the concepts we have learned from lecture. You should read the corresponding lecture notes and Section 2.6, 3.1 - 3.8 of the textbook before getting started. In particular, it's important to read Section 3.5 in detail, since that section covers the cipher that we will break for this project.

In a substitution cipher, each letter is replaced with a different letter. Substitution ciphers commonly appear as word puzzles called cryptograms, which can be solved by recognizing patterns in English words. For example, consider the following cryptogram:

```
LRO KLCZOYLK JY LRJK WNXKK MJNN HO LRO HOKL WIZO HPOXAOPK JY LRO MIPNZ.
```

For substitution ciphers and cryptograms, we commonly write the encoded phrase (called the cipher text) in upper-case letters, and the uncoded phrase (called the plain text) in lower-case letters. This helps us keep track of which is which.

In the cryptogram above, notice that the three letter word `LRO` appears three times in this phrase. This suggests that it's a common English word; some possibilities include `the`, `and`, and `you`. Since `LRO` is the first word in the phrase, it's reasonable to expect that it won't be `and`. Let's guess that `LRO` is `the`, and hope that we get lucky. If `LRO` is `the`, then the letter `L` stands for `t` throughout the text, and `R` for `h`, and `O` for `e`. Making these substitutions, we have

```
the KtCZeYtK JY thJK WNXKK MJNN He the HeKt WIZe HPeXAePK JY the MIPNZ.
```

Next, let's look at the two letter word `JY`. The word `JY` occurs before `the`, and before `thJK`. Thinking about common two letter words that make sense before `the`, `JY` might be `if`, `in`, `is`, `of`, or `on`. Since `J` occurs in the word `thJK`, `J` is probably `i` rather than `o`, and `thJK` is probably `this`. So, we'll replace `J` with `i` and `K` with `s`.

```
the stCZeYts iY this WNXss MiNN He the Hest WIZe HPeXAePs iY the MIPNZ.
```

Next, let's look at the phrase `He the Hest`. From this, we can guess that `H` is `b`. Also, looking at `Y` in the second and third words, `Y` is most likely `n`.

```
the stCZents in this WNXss MiNN be the best WIZe bPeXAePs in the MIPNZ.
```

Now, being excellent pupils, you may be able to guess that `stCZents` is `students`, so we replace `C` with `u` and `Z` with `d`.

```
the students in this WNXss MiNN be the best WIde bPeXAePs in the MIPNd.
```

Then, we can guess that `WNXss` is `class`, based on the context about students. So, we make those replacements.

```
the students in this class Mill be the best cIde bPeXAePs in the MIPld.
```

With a few more guesses, we can arrive at the plain text.

```
the students in this class will be the best code breakers in the world.
```

Typically when solving a cryptogram, you might make some guesses that don't work out, so you might need to back track and try some different possibilities.

For this project, we will write functions that will help you make good substitutions easily, so you can use these functions to solve cryptograms more quickly than you could by hand.

You are to do this project with a partner. Because this project is more difficult than your previous projects, we strongly recommend that you spend some time planning out your functions, before writing any actual code.

Task 1: Recommending Substitutions for Short Words

A common strategy for solving cryptograms is to start by looking for short words. To do this, we first need to separate a cryptogram into individual words. Luckily, we don't have to define this ourselves! There is a string method called `split`, which will accomplish this task for us. You can read about this function in the python documentation: `https://docs.python.org/2/library/stdtypes.html#string-methods`.
Now, your first task is to define a function called `suggest_substitutions` which takes the ciphertext for a cryptogram, and suggests substitutions for the one, two, and three letter words.

- Here are the one letter words to suggest: a, i. We make "i" lowercase, because the plaintext will be all lowercase.
- Here are the two letter words to suggest: of, to, in, it, is, be, as, at, so, we, he, by, or, on, do, if, me, my, up, an, go, no, us, am.
- Here are the three letter words to suggest: the, and, for, are, but, not, you, all, any, can, had, her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way, who, boy, did, its, let, put, say, she, too, use.

On the next page, we give an example of how this function should work, for the following ciphertext.

```
TL LEJGX E PRGEL CGEW KA YREUGRN LK XLEVC HB LK KHR GVGQTGX YHL SHXL EX QHFD
                                LK XLEVC HB LK KHR ARTGVCX
```

```
>>> ciphertext = "TL LEJGX E PRGEL CGEW KA YREUGRN LK XLEVC HB LK KHR GVGQTGX YHL SHXL EX
QHFD LK XLEVC HB LK KHR ARTGVCX"
>>> suggest_substitutions(ciphertext)
One letter words:
E
Suggestions for one letter words: a, i
Two letter words:
TL
KA
LK
HB
LK
EX
LK
HB
LK
Suggestions for two letter words: of, to, in, it, is, be, as, at, so, we, he, by, or, on,
do, if, me, my, up, an, go, no, us, am
Three letter words:
KHR
YHL
KHR
Suggestions for three letter words: the, and, for, are, but, not, you, all, any, can, had,
her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way,
who, boy, did, its, let, put, say, she, too, use
```

The text printed by `suggest_substitutions` should exactly match the formatting of the example above (except the lines that run over to the next line). You may assume that the only punctuation in the text will be commas and periods.

Define your `suggest_substitutions` function in the file `cryptanalysis.py`, which is posted on moodle.

Task 2: Making Substitutions

Now that we've got a way to suggest substitutions for a given ciphertext, we need to figure out how to make a substitution.
To do this, you first need to define a function that determines if a given substitution is valid. For example,

- `ASDF` can't decode to `wordy`, since they have a different number of letters.
- `TYY` can't decode to `and`, since then `Y` would need to stand for both `n` and `d`.
- `GHJK` can't decode to `that`, since then both `G` and `K` would represent `t`.
- `BNM I` can't decode to `is it`, since the spaces don't match up.

This function should be called `is_valid`, and it should take an encoded string and a decoded string, returning `True` if the given substitution is valid, and return `False` if it is not valid. Below, we give an example of how this function should work for various inputs.

```
>>> is_valid("ASDF", "wordy")
False
>>> is_valid("ASDF", "this")
True
>>> is_valid("TYY", "and")
False
>>> is_valid("TYY", "all")
True
>>> is_valid("GHJK", "that")
False
>>> is_valid("GHJK", "what")
True
>>> is_valid("BNM I", "is it")
False
>>> is_valid("BNM I", "and i")
True
```

Note: sometimes cryptograms have the rule that a letter cannot decode to itself. For our cryptograms, we will not have this rule.

When writing your `is_valid` function, it may be tempting to try to write code to test for all of the possibilities given above, before doing any testing. Resist this temptation! We strongly recommend that you try to accomplish one of these checks at a time, and testing it, before moving on to the next piece. For example, first get your `is_valid` function to compare lengths, and test this out, before having the function do anything else. This is best done by defining several helper functions that `is_valid` will call; for example, you might want to define a function `same_length` that takes two strings, and determines if they have the same length. We have defined the `same_length` function for you, and shown how it can be called within `is_valid`. You will need to determine what helper functions you'll need, implement them, and finish writing the body of `is_valid`. All of these functions should include contracts and docstrings.

Define your `is_valid` function, along with your helper functions, in the file `cryptanalysis.py`.

Now, we can use the `is_valid` function in a substitution function, `substitute`. This function takes a cipher text, an encoded string, and a decoded string, and makes the given substitutions in the cipher text. We've provided this function for you, in the file called `cryptanalysis.py`. However, you will need to write a contract and docstring. Here are some examples of how this function can be used, on the same ciphertext from Task 1.

```
>>> ciphertext = "TL LEJGX E PRGEL CGEW KA YREUGRN LK XLEVC HB LK KHR GVGQTGX YHL SHXL EX
QHFD LK XLEVC HB LK KHR ARTGVCX"
>>> substitute(ciphertext, "HB LK", "it is")
'Substitution is invalid'
>>> substitute(ciphertext, "E", "i")
'TL LiJGX i PRGiL CGiW KA YRiUGRN LK XLiVC HB LK KHR GVGQTGX YHL SHXL iX QHFD LK XLiVC HB
LK KHR ARTGVCX'
>>> substitute(ciphertext, "E", "a")
'TL LaJGX a PRGaL CGaW KA YRaUGRN LK XLaVC HB LK KHR GVGQTGX YHL SHXL aX QHFD LK XLaVC HB
LK KHR ARTGVCX'
>>> substitute(ciphertext, "E LK", "i to")
'Tt tiJGX i PRGit CGiW oA YRiUGRN to XtiVC HB to oHR GVGQTGX YHt SHXt iX QHFD to XtiVC HB
to oHR ARTGVCX'
>>> substitute(ciphertext, "E LK KHR", "i to our")
'Tt tiJGX i PrGit CGiW oA YriUGrN to XtiVC uB to our GVGQTGX Yut SuXt iX QuFD to XtiVC uB
to our ArTGVCX'
```

Note that the function `substitute` does not change the value of `ciphertext`; rather, it returns a copy of `ciphertext`, with some of the characters changed.

Task 3: Solving Cryptograms

Finally, it's time to put your cryptanalysis functions to the test! For each of the cryptograms below, use your `suggest_substitutions` function to print a list of suggested substitutions. Then, start using the `substitute` functions to try some substitutions. It might take a few different guesses before you start to get on the right track; fortunately, the `substitute` functions allows you to quickly try substitutions, which would be more time-consuming by hand. Here's an example of using these functions to solve the cryptogram from Task 1.

```
>>> ciphertext = "TL LEJGX E PRGEL CGEW KA YREUGRN LK XLEVC HB LK KHR GVGQTGX YHL SHXL EX
QHFD LK XLEVC HB LK KHR ARTGVCX"
>>> suggest_substitutions(ciphertext)
One letter words:
E
Suggestions for one letter words: a, i
Two letter words:
TL
KA
LK
HB
LK
EX
LK
HB
LK
Suggestions for two letter words: of, to, in, it, is, be, as, at, so, we, he, by, or, on,
do, if, me, my, up, an, go, no, us, am
Three letter words:
KHR
YHL
KHR
Suggestions for three letter words: the, and, for, are, but, not, you, all, any, can, had,
her, was, one, our, out, day, get, has, him, his, how, man, new, now, old, see, two, way,
who, boy, did, its, let, put, say, she, too, use
```

Notice that the two letter word `LK` shows up a lot; let's try to guess what it is. We'll try a few possibilities, and see what looks best.

```
>>> substitute(ciphertext, "LK", "to")
'Tt tEJGX E PRGEt CGEW oA YREUGRN to XtEVC HB to oHR GVGQTGX YHt SHXt EX QHFD to XtEVC HB
to oHR ARTGVCX'
>>> substitute(ciphertext, "LK", "it")
'Ti iEJGX E PRGEi CGEW tA YREUGRN it XiEVC HB it tHR GVGQTGX YHi SHXi EX QHFD it XiEVC HB
it tHR ARTGVCX'
>>> substitute(ciphertext, "LK", "of")
'To oEJGX E PRGEo CGEW fA YREUGRN of XoEVC HB of fHR GVGQTGX YHo SHXo EX QHFD of XoEVC HB
of fHR ARTGVCX'
>>> substitute(ciphertext, "LK", "so")
'Ts sEJGX E PRGEs CGEW oA YREUGRN so XsEVC HB so oHR GVGQTGX YHs SHXs EX QHFD so XsEVC HB
so oHR ARTGVCX'
```

Looking at the other two letter words, it seems most likely that `LK` is `to`, and it seems like `TL` might be `it`. We'll try that substitution.

```
>>> substitute(ciphertext, "LK TL", "to it")
'it tEJGX E PRGEt CGEW oA YREUGRN to XtEVC HB to oHR GVGQiGX YHt SHXt EX QHFD to XtEVC HB
to oHR ARiGVCX'
```

Now, we have a one letter word `E`, which is probably either `a` or `i`. We'll try these.

```
>>> substitute(ciphertext, "LK TL E", "to it i")
'Substitution is invalid'
>>> substitute(ciphertext, "LK TL E", "to it a")
'it taJGX a PRGat CGaW oA YRaUGRN to XtaVC HB to oHR GVGQiGX YHt SHXt aX QHFD to XtaVC HB
to oHR ARiGVCX'
```

So, `E` is probably `a`. Now, `KA` is probably `of`, and `EX` is probably `as` or `at`, we'll try these.

```
>>> substitute(ciphertext, "LK TL E KA EX", "to it a of as")
'it taJGs a PRGat CGaW of YRaUGRN to staVC HB to oHR GVGQiGs YHt SHst as QHFD to staVC HB
to oHR fRiGVCs'
>>> substitute(ciphertext, "LK TL E KA EX", "to it a of at")
'Substitution is invalid'
```

Now, the word `KHR` is probably `one`, `our`, or `out`.

```
>>> substitute(ciphertext, "LK TL E KA EX KHR", "to it a of as one")
'it taJGs a PeGat CGaW of YeaUGeN to staVC nB to one GVGQiGs Ynt Snst as QnFD to staVC nB
to one feiGVCs'
>>> substitute(ciphertext, "LK TL E KA EX KHR", "to it a of as our")
'it taJGs a PrGat CGaW of YraUGrN to staVC uB to our GVGQiGs Yut Sust as QuFD to staVC uB
to our friGVCs'
>>> substitute(ciphertext, "LK TL E KA EX KHR", "to it a of as out")
'Substitution is invalid'
```

The phrase "up to our" makes more sense than the phrase "no to one", so we'll guess that `HB` is `up`. Then, we can also guess that `XLEVC` is `stand`. From here, we can make quick progress.

```
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC", "to it a of as our up stand")
'it taJGs a PrGat dGaW of YraUGrN to stand up to our GnGQiGs Yut Sust as QuFD to stand up
to our friGnds'
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC YHL", "to it a of as our up stand b
ut")
'it taJGs a PrGat dGaW of braUGrN to stand up to our GnGQiGs but Sust as QuFD to stand up
to our friGnds'
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC YHL LEJGX", "to it a of as our up s
tand but takes")
'it takes a Preat deaW of braUerN to stand up to our eneQies but Sust as QuFD to stand up
to our friends'
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC YHL LEJGX P W U N", "to it a of as
our up stand but takes g l v y")
'it takes a great deal of bravery to stand up to our eneQies but Sust as QuFD to stand up
to our friends'
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC YHL LEJGX P W U N Q S", "to it a of
as our up stand but takes g l v y m j")
'it takes a great deal of bravery to stand up to our enemies but just as muFD to stand up
to our friends'
>>> substitute(ciphertext, "LK TL E KA EX KHR HB XLEVC YHL LEJGX P W U N Q S FD", "to it a
of as our up stand but takes g l v y m j ch")
'it takes a great deal of bravery to stand up to our enemies but just as much to stand up
to our friends'
```

Here are the cryptograms for you to solve using your functions. Once you've solved the cryptograms, copy and paste the decoded text into the template file `cryptograms.txt`, which has been provided for you.

These cryptograms are also included in the template file. We strongly recommend that you copy and paste the cryptograms, rather than attempting to transcribe them. This will help you avoid errors.

Cryptogram 1

TAFM QGBPMU FXTPGKBQUM TAU HUPMHUQTFVUM DXK OUTAGKM GJ QGOHBTUP MQFUXQU. MZOLGCFQ FXJGPODTFGX FM PUHPUMUXTUK DM KDTD, WAUTAUP FT LU XBOLUPM, TUIT, GP FODNUM. DBTGODTUK HPGQUMMUM JGP GHUPDTFXN GX TAU KDTD DPU PUHPUMUXTUK LZ NUXUPDC HPGQUKBPUM YXGWX DM DCNGPFTAOM. TAGMU DCNGPFTAOM DPU WPFTTUX FX D HDPTFQBCDP XGTDTFGX, D HPGNPDOOFXN CDXNBDNU, DM HPGNPDOM. MTBKUXTM WFCC CUDPX AGW TG QDPPZ GBT TAUMU TDMYM DXK AGW TG TAFXY DLGBT QGOHBTDTFGX FX TUPOM GJ NUXUPDC HDTTUPXM, MBQA DM AFUPDPQAFQDC QGOHGMFTFGX GP TAU BMU GJ FXTUPQADXNUDLCU QGOHGXUXTM WFTA QGXMFMTUXT FXTUPJDQUM.

Cryptogram 2

GO G EQUUSCJWZ QY OEPQXGRO, WPT YGESXWZ GCA OWSATCWO QY LSOWGMSO GAQXBPSO EQXXTLT PGMT YQRUSXGWTA GC GEGATUJE PQCTOWZ BQXJEZ GCA PQCQR EQAT OZOWTU, KPJEP JO BRJCWTA JC WPT GEGATUJE ISXXTWJC GCA JC WPT LSOWGMSO LSJAT. GO G OWSATCW GW LSOWGMSO GAQXBPSO EQXXTLT, J GLRTT WQ SBPQXA WPT PQCQR EQAT. WPJO UTGCO WPGW J KJXX GIJAT IZ WPT GEGATUJE PQCTOWZ BQXJEZ, GCA GIJAT IZ ATEJOJQCO QY WPT HQJCW OWSATCW/YGESXWZ PQCQR IQGRA. QC UZ PQCQR, J BXTALT

```
WPGW J PGMT CQW LJMTC, RTETJMTA, QR WQXTRGWTA QWPTRO SOT QY SCGSWPQRJDTA GJA
                        JC EQUBXTWJCL WPJO KQRN.
```

Once you have solved the cryptograms, copy the decoded text into the corresponding places in the file `cryptograms.txt`.

Extra Credit: Frequency Analysis

There is extra credit available on this project, which is to be completed and submitted individually. This task is described in a separate document.

Submitting your work

You will be submitting your code and answers using Moodle. For this project, you will need to submit the following files:

- `cryptanalysis.py`, which is the Python file containing the functions `suggest_substitutions`, `is_valid`, helper functions for `is_valid`, and `substitute`.
- `cryptograms.txt`, which is a text file containing your solutions for the cryptograms.

Because you are doing this project in pairs, we only require one submission per pair. Be sure to include the names of both partners in the file `cryptograms.txt`.

Grading

You will earn one point for each of the following accomplishments:

- You have written a correct contract and docstring for the `suggest_substitutions` function.
- Your `suggest_substitutions` function correctly divides the given ciphertext into individual words.
- Your `suggest_substitutions` function uses the `len` function.
- Your `suggest_substitutions` function correctly identifies one, two, and three letter words.
- Your `suggest_substitutions` function correctly prints suggestions for one, two, and three letter words.
- Your `suggest_substitutions` function produces output with formatting matching the given example.
- You have written a correct contract and docstring for the `is_valid` function, as well as its helper functions.
- Your `is_valid` function uses control flow and/or a boolean expression to return a boolean value.

- You have defined helper functions for your `is_valid` function.
- Your `is_valid` function is concise, and relies primarily on helper functions.
- Your `is_valid` function returns `False` if it receives two strings with different lengths.
- Your `is_valid` function returns `False` if the spaces of the input words don't match up.
- Your `is_valid` function returns `False` if the given substitution would require two different letters to decode to the same letter.
- Your `is_valid` function returns `False` if the given substitution would require one letter to decode to two different letters.
- Your `is_valid` function returns `True`, if none of the above conditions hold.
- You have written a correct contract and docstring for the `substitute` function.
- (2 points) You correctly solve Cryptogram 1.
- (2 points) You correctly solve Cryptogram 2.