HW10: Ciphers

- Assignment HW10 should be completed individually.
- Be sure to read the entire prompt and understand the problem before beginning coding.

Learning Objectives

- Work with both input and output files.
- Working with classes, particularly creating objects and calling methods.
- Explore ciphers, encrypting, and decrypting.
- Understanding how to handle merge conflicts

How to Start

- To begin, make a copy of this document by going to File >> Make a Copy...
- Share the copied document with all members of your team, if you are working with someone. You can share this document by hitting the blue button in the top right of the document, then entering the email address of all members in the bottom input field.
- Change the file name of this document to ***username1* - HW10: Ciphers** (for example, **heggens - HW10: Ciphers**). To do this, click the label in the top left corner of your browser.
- Next, go to the **Github classroom for HW10**.
- Paste the link to your Github repo here:

Github Repo Link:	`https://github.ceom/Berea-College-CSC-226/hw10-caesar-ciphers-BesherKitazBerea`

- Open PyCharm. If you are not at the Welcome to PyCharm page, go to File >> Close project.
- Using the **Get from VCS** button, open your repository using the URL above.
- Create a new branch to store your work on.

Translation algorithms used to crack centuries-old code

Read the following article published on 25 October 25, 2011, by Mark Brown

From: https://www.wired.com/2011/10/copiale-cipher-crack/

> Computer scientists from Sweden and the United States have applied modern-day, statistical translation techniques -- the sort that are used in Google Translate -- to decode a 250-year old secret message.
>
> The original document, nicknamed the Copiale Cipher, was written in the late 18th century and found in the East Berlin Academy after the Cold War. It's since been kept in a private collection, and the 105-page, slightly yellowed tome has withheld its secrets ever since.
>
> But in 2011, University of Southern California Viterbi School of Engineering computer scientist Kevin Knight -- an expert in translation, not so much in cryptography -- and colleagues Beáta

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

Megyesi and Christiane Schaefer of Uppsala University in Sweden, tracked down the document, transcribed a machine-readable version and set to work cracking the centuries-old code.

The book's pages -- bound in gold and green brocade paper -- contained about 75,000 characters in very neat handwriting. Outside of two words -- an owner's mark ("Philipp 1866") and a note in the end of the last page ("Copiales 3") -- the rest was encoded.

Some of the letters were obviously Roman and others were plainly Greek, while the rest were abstract symbols and doodles.

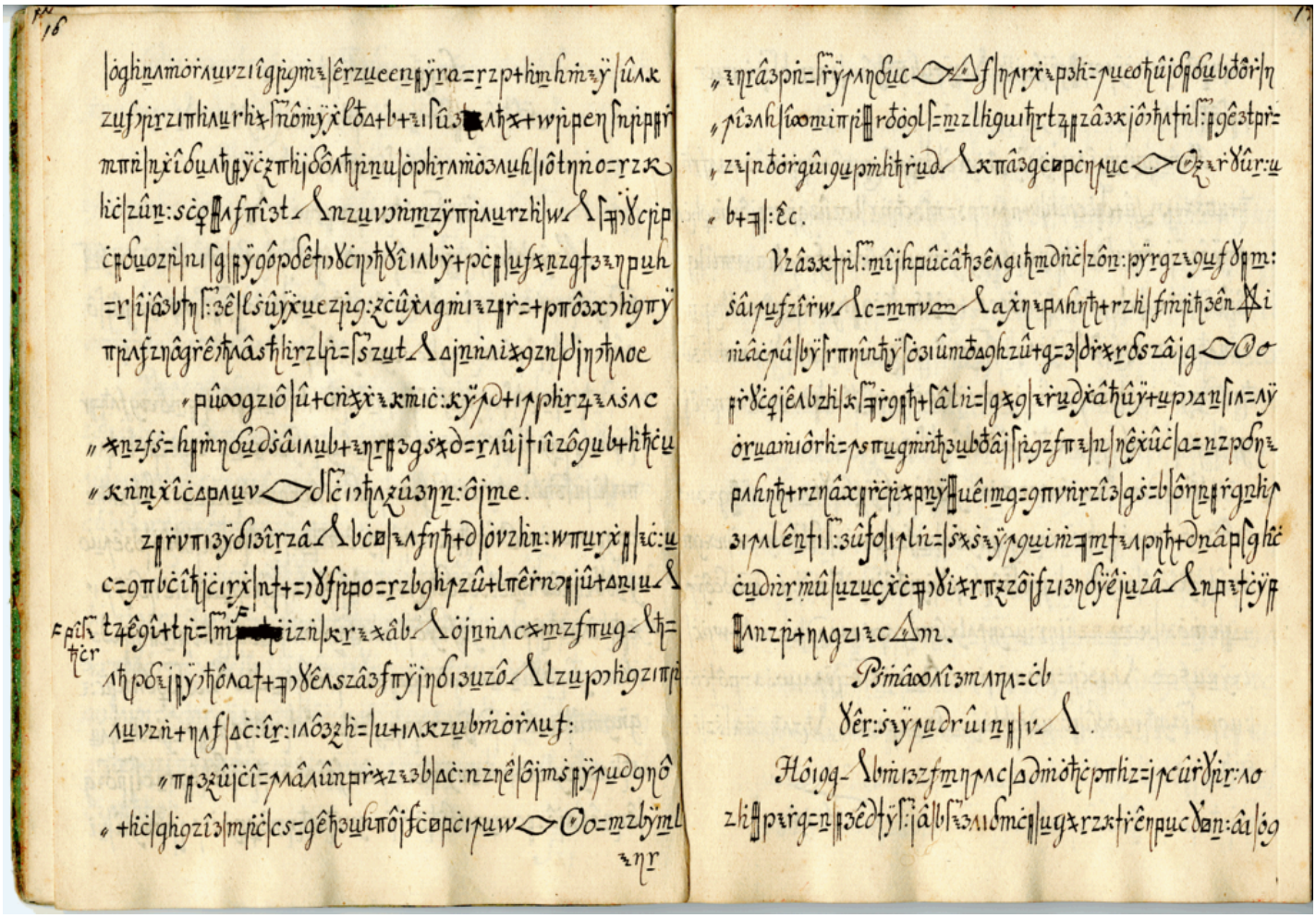
At first, Knight and his team isolated the Roman and Greek characters, figuring that they might be the real message, and attacked it with a home-made translation project. 80 different languages, and many hours later, and nothing happened. "It took quite a long time and resulted in complete failure," says Knight.

The team realized that the known characters were just there to mislead. So they booted them out and looked at the symbols. They theorized that abstract symbols with similar shapes might represent the same letter, or groups of letters. They tested this with different languages and when German was used, some meaningful words emerged -- "Ceremonies of Initiation", followed by "Secret Section".

A little computation later and a good chunk of the book had been decoded and transcribed. The document revealed the rituals and political leanings of a German secret society, and one that had a strange obsession with eyeballs, plucking eyebrows, eye surgery and ophthalmology. You can read the entire, weird, manifesto in English at
http://stp.lingfil.uu.se/~bea/copiale/copiale-translation.txt

Buoyant from his success, Knight is now planning on using his techniques and programs to tackle other codes including ones from the Zodiac Killer, a Northern Californian serial murderer from the 60s; "Kryptos," an encrypted message carved into a granite sculpture on the grounds of CIA headquarters; and the Voynich Manuscript, a medieval document that has baffled professional cryptographers for decades.

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License



You may optionally see the entire manuscript describing the work at http://www.isi.edu/natural-language/people/copiale-11.pdf.

Go to **hw10_ciphers_questions.md** and answer **SECTION 1**, then come back to this document to complete the rest of the assignment

---

Resumes - Part II

In HW08, we asked you to write your resume. In this homework, we will expand that resume. As you notice from the reading above, computer science is a field that spans across many other disciplines; in this case, history, language arts, and archeology. In this assignment, we want *you* to look forward. Imagine yourself in 10 years... how are you using computer science? Are you interested in security? architecture? political science? How do your personal interests intersect with your career aspirations?

Go to **hw10_ciphers_questions.md** and answer **SECTION 2**, then come back to this document to complete the rest of the assignment

---

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

Instructions

1. Make a copy of your resume. Hopefully, you took my advice and created a Google doc, which will make this step super easy!
2. This new version of your resume is *aspirational*. Revise your resume to represent the professional you in 10 years, after you've graduated and established yourself in the tech industry (or another field!).
3. Start with year 10. What is the dream job that you want to have after 10 years of hard work? You do not need to include a company name (though, it is not forbidden, either). You DO need to include a job title, as well as two or more bullet points about what your duties are in that role. For example, if your dream job is "Associate Professor of Computational Archaeology", a bullet point about duties would be "Developing algorithms for decrypting strange ancient manuscripts about eyes". The more specific, the better!
4. Work backwards from year 10, writing an entry for the previous two years. What job led you to that year 10 dream job? Keep working backwards every two years until you reach year 0 (i.e., you, right now!). If you get stuck working backwards, switch tactics and start at year 0, and work forwards in time.
5. The goal here is to have a 10-year plan, with milestones every two years for how you are advancing towards that goal. If your ten years has gaps or holes (for example, a jump from junior software engineer to project manager), fill them! Remember, this is an aspirational document!

Caesar Ciphers

Encryption has been in the news a great deal recently:

- July 11, 2013: Microsoft handed the NSA access to encrypted messages
- December 21, 2013: Report: NSA Paid RSA $10M to Create 'Back Door' in Encryption Software
- March 12, 2014: Google is encrypting search globally. That's bad for the NSA and China's censors
- February 17, 2016: Apple, The FBI And iPhone Encryption: A Look At What's At Stake
- March 27, 2018: FBI has a unit solely devoted to its 'going dark' problem
- November 3, 2023: Scientist Claims Quantum RSA-2048 Encryption Cracking Breakthrough

The Google article proclaims the following:

"Google has begun routinely encrypting Web searches conducted in China, posing a bold new challenge to that nation's powerful system for censoring the Internet and tracking what individual users are viewing online.

The company says the move is part of a global expansion of privacy technology designed to thwart surveillance by government intelligence agencies, police and hackers who, with widely available tools, can view e-mails, search queries and video chats when that content is unprotected."

Encryption and privacy are clearly big news.

Suppose you intercepted an email between two people that was using some kind of encryption, thereby creating a cryptogram:

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

> *pyhv ngyvr owf nrjrw urovn omy yhv posbrvn qvyhmbs pyvsb yw sbxn gywsxwrws, o wrt wosxyw, gywgrxjrf xw kxqrvsu, owf frfxgosrf sy sbr dvydynxsxyw sbos okk lrw ovr gvrosrf rzhok.*

> *wyt tr ovr rwmomrf xw o mvros gxjxk tov, srnsxwm tbrsbrv sbos wosxyw, yv owu wosxyw ny gywgrxjrf owf ny frfxgosrf, gow kywm rwfhvr. tr ovr lrs yw o mvros qosskr-pxrkf yp sbos tov. tr bojr gylr sy frfxgosr o dyvsxyw yp sbos pxrkf, on o pxwok vrnsxwm dkogr pyv sbynr tby brvr mojr sbrxv kxjrn sbos sbos wosxyw lxmbs kxjr. xs xn oksymrsbrv pxssxwm owf dvydrv sbos tr nbyhkf fy sbxn.*

> *qhs, xw o kovmrv nrwnr, tr gow wys frfxgosr -- tr gow wys gywnrgvosr -- tr gow wys bokkyt -- sbxn mvyhwf. sbr qvojr lrw, kxjxwm owf frof, tby nsvhmmkrf brvr, bojr gywnrgvosrf xs, pov oqyjr yhv dyyv dytrv sy off yv frsvogs. sbr tyvkf txkk kxsskr wysr, wyv kywm vrlrlqrv tbos tr nou brvr, qhs xs gow wrjrv pyvmrs tbos sbru fxf brvr. xs xn pyv hn sbr kxjxwm, vosbrv, sy qr frfxgosrf brvr sy sbr hwpxwxnbrf tyvc tbxgb sbru tby pyhmbs brvr bojr sbhn pov ny wyqku ofjowgrf. xs xn vosbrv pyv hn sy qr brvr frfxgosrf sy sbr mvros sonc vrloxwxwm qrpyvr hn -- sbos pvyl sbrnr bywyvrf frof tr socr xwgvronrf frjysxyw sy sbos gohnr pyv tbxgb sbru mojr sbr kons phkk lronhvr yp frjysxyw -- sbos tr brvr bxmbku vrnykjr sbos sbrnr frof nbokk wys bojr fxrf xw joxw -- sbos sbxn wosxyw, hwfrv myf, nbokk bojr o wrt qxvsb yp pvrrfyl -- owf sbos myjrvwlrws yp sbr drydkr, qu sbr drydkr, pyv sbr drydkr, nbokk wys drvxnb pvyl sbr rovsb.*

Looks like gibberish, right? Actually, what was used was a simple cipher-substitution in which each letter ("a" through "z") in the original text was replaced by a different, random one. The cryptogram above, for example, has had every occurrence of the letter 'e' replaced with an 'r', every occurrence of 'w' by a 't', and so forth. The word "tr", which is the sixth word in the last paragraph, was therefore originally "we". The trick is to find which letter substitution was used and you can then reverse the process to get the original message.

Though not the first cipher, one of the simplest and most notable ciphers is the Caesar cipher, used by Julius Caesar to send messages to his generals during war. The cipher was simple; each letter is simply shifted in the alphabet by a number, which Caesar and the general would agree upon ahead of time. So a shift of +3 would equate to 'a' becoming 'd', 'b' would become 'e', and so on.

The **hw10_caesar_cipher.py** code demonstrates a Caesar cipher in action, using **classes**. Below is a CRC card, which represents the class. A CRC card is typically done on a notecard (hence, the card name). The point of a CRC card is to summarize all of the aspects of a class: the class attributes, the class methods, a description of what they are all used for, as well as any class collaborations.

Read the CRC card for the CaesarCipher class, and try to map the ideas to the code.

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

CRC Card for the CaesarCipher class

Class name: CaesarCipher	
Instance Attributes:	**Class Collaborations (other classes):**

• `alphabet`:
  ○ Class variable containing the entire alphabet, in order. Used to do our shifts.
• `self.input_file`:
  ○ the file to be encrypted or decrypted
• `self.key`:
  ○ an integer representing the amount each message/cipher will be shifted
• `self.message`:
  ○ holds the message
• `self.cipher`:
  ○ holds the cipher
• `self.crypt_type`:
  ○ holds either "encrypt" or "decrypt"
	• None

Class Methods:	**Class Collaborations (other classes):**

• `__init__()`:
  ○ a constructor for the CaesarCipher class
• `import_file()`:
  ○ imports a file stored in the variable `self.input_file` and returns a string representing the contents of the file
• `export_file()`:
  ○ exports the message/cipher to a file
• `encrypt()`:
  ○ converts an original message into a ciphered message which is returned with each letter shifted to the right by the key
• `decrypt()`:
  ○ converts a ciphertext and returns the original message which is found by shifting each letter to the left by the key
	• None

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

The Instructions

First, read and understand the **hw10_caesar_cipher.py** code.

Then, modify the **hw10_caesar_cipher.py** file to complete the following tasks:

1. Caesar has two letters to send: ***letter_to_friend_1.txt*** and ***letter_to_friend_2.txt.*** Complete the code in `main()` to encrypt the two messages and generate two new encrypted files:
    a. ***cipher_to_friend_1.txt***
    b. ***cipher_to_friend_2.txt***
2. The `CaesarCipher` class is incomplete; while the program can encrypt a message, the program should also be able to decrypt the message. Complete the following tasks related to decryption:
    a. Complete the `decrypt()` function for decrypting the message inside the `CaesarCipher` class. Pay attention to the `encrypt()` method for hints at how to decrypt (i.e., it should "undo" the encryption process).
    b. Once you've completed the decrypt function, make the correct calls in the `main()` function to construct a new CaesarCipher object and decrypt the message. We've included a ciphered letter for testing with, called ***cipher_from_friend_3.txt***
    c. Use the `export_file()` function to write your decrypted message to a file named ***message_from_friend_3.txt***
    d. Note that your code should be able to decrypt *any* letter which was encrypted using the Caesar cipher logic.

Merge Conflicts

Now that we've grown comfortable with using git and can successfully avoid conflicts, let's grasp what to do when we inevitably have a real conflict. While merge conflicts sound scary, they are in actuality a protection mechanism. They prevent two programmers from editing the same lines of code in any file. While good communication will avoid merge conflicts, sometimes they are unavoidable and must be handled by you, the programmer.

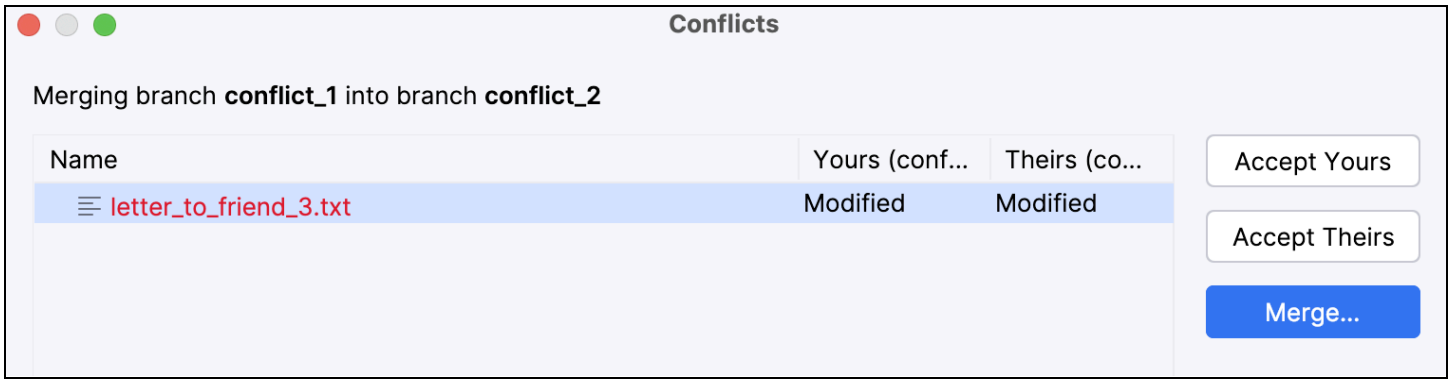
In this exercise, we will simulate a merge conflict between yourself and… yourself! Here is the basic premise:

1. Create a new branch from your current working branch. Let's call this branch *conflict_1*.
2. Create a file called **letter_to_friend_3.txt.** Add a message.
3. Commit your changes (a push isn't necessary).
4. Go back to main current working branch from step 1.
5. Create a new branch from your current working branch again, called *conflict_2.*
6. Create another file called **letter_to_friend_3.txt**. Add a unique message.
7. Commit your changes (a push isn't necessary).

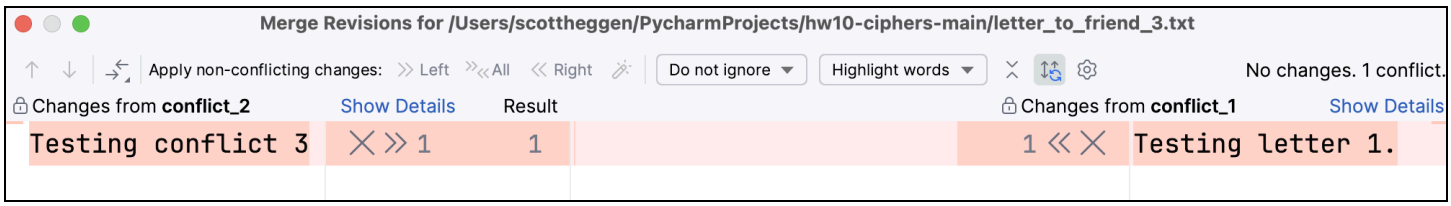
Hopefully at this point you are clearly seeing the conflict: two unique branches with the same filename, holding different content. Let's now do the merge!

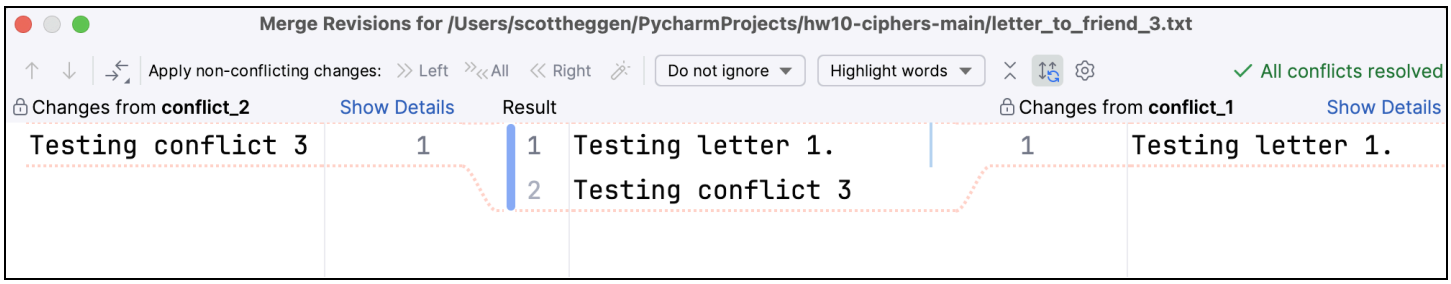
1. Go to the *Git* menu, and select *Merge…*
2. Select the other branch (i.e., if you're still on *conflict_2*, select *conflict_1*).
3. Pycharm should notice the conflict, and ask you to make a choice: *Accept Theirs*, *Accept Yours*, or ***Merge…***. Almost always, you'll select *Merge…* to ensure you are making the right choice.

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License



4. You'll see a screen similar to below, where you now have to make choices. On the left is the "code" in your current branch. On the right is the "code" in the branch that you are merging into your current branch. In the center is the code you want to keep. In this case, keep both codes by hitting the triple arrow on both sides, as shown below:





5. When you have completed your merge conflict resolution, you'll notice there is another commit added to your commit history showing the merge was successful. That's it!

When you have successfully resolved the merge conflict on your conflict_1 or conflict_2 branch, switch back to your current working branch with the ciphers assignment complete, and merge your conflict_x branch into it following the same process. If you have merge conflicts, feel confident you can resolve them now by reviewing the code and making wise choices!

> Go to **hw10_ciphers_questions.md** and answer **SECTION 3**, then come back to this document to complete the rest of the assignment

---

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License

Submission Instructions

Follow the submission instructions to submit your work by Wednesday at 11:55PM.

Copyright © 2023 | Licensed under a Creative Commons Attribution-Share Alike 3.0 United States License