# MSOE EECS Department
# CS2911: Week 2 Lab Grading Checklist
Dr. Yoder Name:

| Item | Points |
|---|---|
| Introduction: Describe the lab in your own words. (You may use the space below.) | / 1 |
| Problems 1 and 8. (See requirements in exercises) | / 1 |
| Problems 2 and 9 | / 1 |
| Problems 3 and 10 | / 1 |
| Problems 4 and 11 | / 1 |
| Problems 5 and 12 | / 1 |
| Problems 6 and 13 | / 1 |
| Problem 14. Write the two types you found. | / 1 |
| Problems 15 and 16. Do not use str, int, etc. | / 1 |
| Problems 17 and 18. Do not use str, int, etc. | / 1 |
| Problems 19 and 20. Do not use str, int, etc. | / 1 |
| Problems 21 and 22 | / 1 |
| Problem 23 | / 1 |
| Problems 24 and 25 | / 1 |
| Problems 26 and 27 | / 1 |
| Problem 28 | / 1 |
| Problem 29. Be creative and imaginative. | / 1 |
| Summarize what you learned during this lab. (You may use this space.) | / 2 |
| Things you liked or suggestions for improvement (Required; you may use this space) | / 1 |
| **Total** | / 20 |

**Follow these instructions for full credit:**

- ***Staple*** this lab cover sheet on top of all the materials you are submitting.
- Submit your work in the ***order*** listed above.
- In addition to the materials above, submit any other supporting materials you create while working the lab where they fit best in your report.
- Your lab packet is due at the start of the following week's lab period. There is a 2 point per day late penalty on the packet. Slip your submitted lab packet under my office door or submit your packet to me during the laboratory.

# Lab 2: Python Encoding

Work through the first few problems on paper before starting Python. ***Box*** your answers. I encourage you not use a calculator, using the space provided or extra paper.

1. Predict how the `bytes` object `b'2 Faced'` will be stored in Python. ***Write*** your answer in hexadecimal shorthand.

2. Predict how the bytes object b'\r\n' will be stored in Python. ***Write*** your answer in both **hexadecimal shorthand** and **bits (binary)**.

3. Predict how the number `104` will be stored in Python. ***Write*** your answer in **binary**, then ***write*** it in **hexadecimal shorthand**.

4. Predict how the number 0xfe19 will be stored in Python. ***Write*** your answer in **hexadecimal shorthand**, then ***write*** it in **binary**.

5. Predict how the bytes object b'\xbeef' will be stored in Python. ***Write*** your answer in **hexadecimal shorthand**, then ***write*** it in **binary**.

6. Predict how the number `1055` will be stored in Python. ***Write*** your answer in **binary**, then ***write*** it in **hexadecimal shorthand**.

7. Set up the `showbits` library: (See lab page for video version of these instructions)
   a. Go to the Lab 2 webpage and download the python module `showbits.py`.
   b. Place the file directly inside the top-level of your Python project.
   c. Open the Python console using Tools -> Python Console.
   d. In the console, type `from showbits import bits, shorthand`. (If you use this in a file, use `import showbits` instead, and use `showbits.bits()` with the package-name when calling `bits()`.)

***As you check*** your answers to your previous problems, either ***write*** a check-mark if the answer was correct, or ***write*** the difference between your prediction and the actual value and ***write*** what you learned from it.

8. ***Check*** your answer to Problem 1 by typing `shorthand(b'2 Faced')`.

9. ***Check*** your answer to Problem 2 by typing `shorthand(b'\r\n')`.

10. ***Check*** your answers to Problem 3 by typing `bits(104)` and `shorthand(104)`.

11. ***Check*** your answers to Problem 4 by typing `shorthand(0xfe19)` and `bits(0xfe19)`.

12. ***Check*** your answers to Problem 5 by typing `shorthand(b'\xbeef')` and `bits(b'\xbeef')`.

13. ***Check*** your answers to Problem 6 by typing `bits(1055)` and `shorthand(1055)`.

14. Set `i = 3`. Use Python to ***determine*** the type of `i`. (See Java/Python table for hints.) Also ***determine*** the type of `int`. ***Write*** the **two** types.

In Problems 15-20, do not use the str, int, hex, format, etc. functions. Instead use the commands introduced in lab. The phrase "Instruct Python to copy" indicates that you should use a Python command that converts an object of one type into another rather than writing a literal value in the new type. There is no need to write your variables in hexadecimal shorthand unless the problem specifies to do so.

15. ***Assign*** the number $1000_{10}$ to a variable. Instruct Python to copy the stored bits contained in this variable into a 16-bit Python **bytes** object. (The bytes object should store the same bits as the int internally.) Looking back at your notes, check that the bytes object has the correct values in it. ***Write*** the Python code you used here:

16. In Problem 9, you created the `bytes` object `b'2 Faced'` by simply typing it into Python. Now, create the `bytes` object by ***specifying*** the hexadecimal shorthand you found in Problem 1 in an `int` literal. For example, to store the hexadecimal shorthand 12 34 FF into a variable, you could type `i=0x1234ff`. Use an `int` literal like `0x`... rather than a bytes object like `b'\x__...'`. You should get an `int` object that, when you look at its bits in hex shorthand, stores your answer to Problem 8.

17. Next, instruct Python to ***copy*** the bits stored in a variable (in Problem 16) into a Python bytes object, just as you did with the number 1000 in Problem 15. ***Display*** the bytes object to check if it is `b'2 Faced'`. ***Write*** the Python code you used just for the transfer here:

18. In Problem 14, you found the hexadecimal shorthand for 1055. ***Pad*** this with zeros (if needed) to create a two-byte number, and write it as a bytes object: `b'\x__ __\x__ __'`. Next, ***copy*** the contents of the bytes object back into an integer. (The `int` should store the same bits as the `bytes` object.) It should be the number 1055. ***Write*** the Python code you used here:

19. Write Python code to store the number 3338 as an `int` and copy its contents into a two-byte `bytes` object. (The `bytes` object should store the same bits as the `int`.) Display this `bytes` object in Python. ***Write*** **what is displayed** and **your Python code** here:

20. Write Python code to store the number 1885889911 as an int and transfer it to a four-byte `bytes` object. Display this `bytes` object in Python. ***Write*** what is displayed and your Python code here: (You can also try 6644322 and 1885888884.). ***Explain*** why these results display the way they do.

21. The command `user_says = input('Please enter the length of the file')` will prompt the user to enter the length of a file, but it returns it as a `str` rather than a Bytes object. ***Write*** some python code to store the actual number as an `int` in the variable `file_length`.

22. Suppose `file_length = 100`. ***Write*** some Python code that produces the bytes object `b'File length: 100'`. The number 100 should change with the file_length variable. (You can concatenate bytes objects just like strings.)

23. Both `to_bytes()` and `encode()` produce a bytes object. ***Describe*** the difference between these methods.

24. ***List*** the basic transformations that `str(i)` uses (if any) to convert the bits stored in the `int` variable `i` to the bits stored in its `str` output.

25. ***List*** the basic transformations that `i.to_bytes(2,'big')` to convert the bits stored in the `int` variable `i` to the bits stored in its `bytes` object output.

26. Considering your answers to Problems 24 and 25, describe why we might expect `str()` to take longer to run than `to_bytes()`.

27. Write a bytes object **literal** holding the raw binary number 500. (That is, write a bytes object value that you could assign to a variable without writing any functions.)

28. Write a bytes object **literal** holding the ASCII decimal number 500.

29. Imagine you just received a bytes object `b` from a sender over the network. The bytes object holds the bits with hexadecimal shorthand `42 41`. If we run the command `int.from_bytes(b,'big')`, we get the python `int` `16961`. If we run the command `b.decode('ASCII')`, we get the python `str` `'BA'`.

    ***Describe*** how you could determine whether it is a raw binary number or ASCII text that is meant to be stored in the `bytes` object. In other words, did the user mean to send the `int` or the `str`? Be creative – there are multiple ways you might know. But you must think outside the bytes object. The goal is critical thinking, not remembering some fact from class or the book.

---

(you may use this space, but I don't expect you will need to)