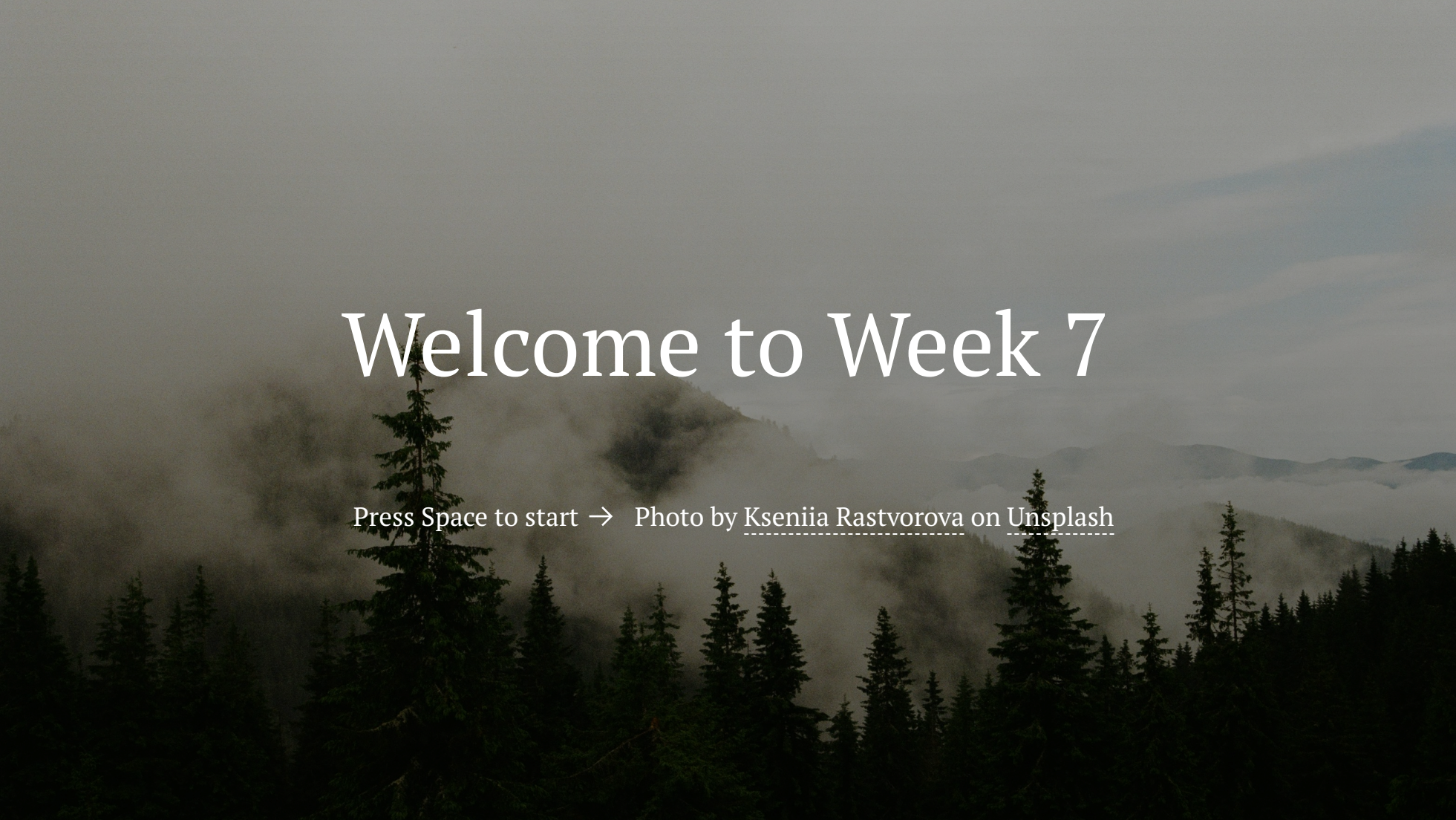

Welcome to Week 7

Press Space to start → Photo by Kseniia Rastvorova on Unsplash

Table of contents

Announcements

- Midterm on Oct 21
- Plan for this thursday:
  - Participation quiz on Thursday (48 hours to complete)
  - Then, review material from the whole semester (focus on exam-style questions)
- Assignment 4 will be slightly delayed + extended for the holiday

Midterm

- Remember to check Canvas for details about exam rules
- My suggestion:
  - First, do the mock exam!
  - Then, take your choice of: review readings, review slides, review extra questions in Canvas

Thursday of this week

- Participation quiz (review concepts from Pizza Party Assignment)
- Then, relatively free form review.
  - I will prioritize your questions. Bring questions!
  - Otherwise, I will work through learning objectives again, and highlight suggestions of content to review and content to put on your notes sheet.

Content

- FAQs from past week
  - Memorization
  - Feeling behind
  - Hold a tiny bit of time for partially solicited studying advice
- Review min and max
- Bits and bytes

Review min and max

Pseudo code?

Pseudo code:

- Initialize a variable to hold max (min). It should be smaller (larger) than items
- Iterate through items
- If item is larger (smaller) thatn current value of variable, store it in our variable
- After all items are seen, we have the max

```
values = [3, 5, 2, 8, 6]
maximum = values[0]
for value in values:
    if value > maximum:
        maximum = value
print(maximum)  # 8
```

How to keep track of the *index* of the max or min?

```
values = [3, 5, 2, 8, 6]
maximum = values[0]
index_of_max = 0
for i in range(len(values)):
    value = values[i]
    if value > maximum:
        maximum = value
        index_of_max = i
print(maximum, i)
```

Quick reminder about this in practice.

If you're prone to exploring, you may have tried typing something like...

```
max([1,2,3])
```

Which works!

- In practice, we'll often use pre-written functions and methods.
- In fact, it's safer to do so. It's scary to write max-finding function from scratch every day if you work in high-stakes setting (factory, energy, transport, medicine...)
- We're doing it in 120 so you understand what's happening under the hood when you call these functions later in life
- Even if you might end up doing a lot of your coding in Excel / spreadsheets (many business contexts)

Binary numbers and ascii

- New content!
- Why learn? It’s how information is stored on our computers

Immersion program starting now:

01001000 01100101 01101100 01101100 01101111 00100000 01010111 01101111 01110010 01101100 01100100

Fun fact – there's a binary message in the building where I studied computing science!
https://newsroom.ucla.edu/stories/a-coded-message-hidden-in-floor-247232

(It was discovered by a student and posted to r/ucla subreddit)

Why binary?

What are (digital) computers?

- Machines that convert low and high electrical signals into 0’s and 1’s
- Then we do some math with the 0’s and 1’s

Bits

A bit is single unit of information that has either the value zero or one

- 0
- 1

(A "1" might correspond to a charged capacitor; in practice, to store more information in a smaller space, we take advantage of transistors, a type of "semiconductor" material that can switch between conducting and insulating)

Bytes

- A byte is 8 bits

END MIDTERM CONTENT HERE.

TLDR: for midterm, understand that under the hood everything is 0s and 1s (false and true. NO problems on midterm related to converting binary to decimal, hexadecimal, interpreting binary code. Also no capacitors or semiconductors, of course!)

Using decimal to represent numbers

In "decimal" (aka numbers you are used to seeing), e.g. 10, 250, 11713 each digit represents powers of ten.

In the number '345'

- the 3 represents 3 * 10^2 = 300, because 10^2=100
- the 4 represents 4 * 10^1 = 40
- The 5 represents 5 * 10^0 = 5

As we add digits to the *left* hand side of our decimal numbers, we get higher powers of ten

Lefthand side is just a convention, by the way! Imagine some mirror world, we decide that

the number 123 is now written as 321. No reason this couldn’t be the case! (Left to right reading is also just a convention).

Using binary to represent numbers

In the binary the bits represents powers of 2

- 1 (2^0)
- 2 (2^1)
- 4 ($2^2 = 2 * 2$)
- 8 ($2^3 = 2 * 2 * 2$)
- 16 ($2^4 = 2 * 2 * 2 * 2$)
- ...

Example of 2 digit binary numbers

- 00 -> 0*1 (rightmost digit)* + *0*2 (2nd from right)= 0
- 01 -> 1*1* + *0*2 = 1
- 10 -> 0*1* + *1*2 = 2
- 11 -> 0*1* + *1*2 = 3

Challenge:

- What is the maximum number we can store with 4 bits
- What about a byte?
- Extreme challenge (trying using your Python terminal): 4 bytes

```python
def max_number_for_bits(n_bits):
    return 2**n_bits - 1

# Example usage:
n_bits = 4
print(f"The maximum number that can be stored with {n_bits} bits is {max_number_for_bits(n_bits)}")
```

- ASCII is a table that maps decimal numbers to characters
- So, if we have a binary number, we can map it to a decimal number and then to a character

Converting binary to decimal in Python

- Any ideas how we'd do it?

```
binary_str = "1101"
decimal = 0
length = len(binary_str)

for i in range(length):
    bit = binary_str[i]
    if bit == '1':
        decimal += 2 ** (length - i - 1)

print(f"The decimal representation of binary {binary_str} is {decimal}")
```

with a few bells and whistles (preview content – don’t need to feel 100% comfortable with this right now)

```
def binary_to_decimal(binary_str):
    decimal = 0
    length = len(binary_str)
    for i, bit in enumerate(binary_str):
        decimal += int(bit) * (2 ** (length - i - 1))
    return decimal
```

Example usage:

```
binary_str = "1101"
print(f"The decimal representation of binary {binary_str} is {binary_to_decimal(binary_str)}")
```

(just fyi)

```python
def binary_to_decimal(binary_str):
    return int(binary_str, 2)

# Example usage:
binary_str = "1101"
print(f"The decimal representation of binary {binary_str} is {binary_to_decimal(binary_str)}")
```

Another "code"

Anyone recognize this?

- #00FFAA
- #FF0000

It’s hexadecimal!

Used for colors, among other uses

Using our recsys example (parallel lists) to handle this new content

```
print("Lets find the most popular coffee shop")

survey_responses = [
    "tims", "sbux", "ren", "tims", "sbux", "blenz",
    "blenz", "ren", "blenz"
]
options = ["tims", "sbux", "ren", "blenz"]

list_of_counting_vars = len(options) * [0]

for response in survey_responses:
    for i in range(len(options)):
        if options[i] == response:
            list_of_counting_vars[i] += 1

for i in range(len(options)):
    print(options[i], list_of_counting_vars[i])
```

With ASCII

```
dec = ["65", "66", "67"]
chars = ["A", "B", "C"]

characters_as_dec = ["65", "65", "65", "67", "67", "67"]

for i in range(len(characters_as_dec)):
    for j in range(len(dec)):
        if characters_as_dec[i] == dec[j]:
            print(chars[j])
```