Discussion 1

DSC 20, Spring 2023

Welcome to DSC 20! 

Agenda

- Why bother learning how to code?
- Basic File Structure
- Loop Overview
- Style - variables
- Practice Activity / python tutor demo
- Class Advice

About the TAs

Benjamin Chen (call me Ben)

- Originally from the Bay Area (San Jose/Fremont)
- Third year DS student minoring in design from Muir
- 5th quarter teaching DSC 20
- Outside the classroom 👨‍🏫: lots of activities (track & field, cross country, badminton, marching band), reading, video games, anime, cooking, eating, travelling, mint addict, bracelet making, skateboarding, 3d renders, video/photo editing, etc.

Charisse Hao (call me Charisse, pronounced "sha-reese")

- Originally from the Bay Area (Fremont)
- Second year DS student minoring in cog sci from Sixth
- 2nd quarter teaching DSC 20
    - Previously taught DSC 10
- Outside the classroom 👨‍🏫: Tennis, dogs + chinchillas, crochet, miniature models, 3d puzzles, gacha games, etc.

Charisse's dog

Why bother learning how to code?

- "gpt-x can do everything"
- "*insert amazing futuristic model here* will replace our job"
- "*insert amazing futuristic model here* will cure cancer"

Scenario

Overpopulation is a looming issue that lots of people are worried about. Clearly, people are not coming up with good enough solutions, so why don't we just ask chatGPT or some other machine learning algorithm and use its solution? What could possibly go wrong?

A pretty realistic machine output to this problem would be the same solution Thanos came to in the MCU - simply remove half of all life and overpopulation is solved. Does this solution work? Sure, it does! Is it one that we would accept? No!! (I hope...)

Problem Statement

Although machine learning only gets better, at the end of the day, it is still a machine. It fails to capture all the qualities that go into complex scenarios. Yes, they are smart and honestly terrifying sometimes, but they are just **tools** at the end of the day.

Problem Statement

Although machine learning only gets better, at the end of the day, it is still a machine. It fails to capture all the qualities that go into complex scenarios. Yes, they are smart and honestly terrifying sometimes, but they are just **tools** at the end of the day.

Without a well versed user, tools are nothing more than scrap. This is where programmers come in. Until the day that machines are truly intelligent, we as programmers are the one who can use them effectively.

Problem Statement

Although machine learning only gets better, at the end of the day, it is still a machine. It fails to capture all the qualities that go into complex scenarios. Yes, they are smart and honestly terrifying sometimes, but they are just **tools** at the end of the day.

Without a well versed user, tools are nothing more than scrap. This is where programmers come in. Until the day that machines are truly intelligent, we as programmers are the one who can use them effectively.

Programming is simply the language that translates human intention to computer actions. It instructs machines on how to carry out your logic.

Problem Statement

Although machine learning only gets better, at the end of the day, it is still a machine. It fails to capture all the qualities that go into complex scenarios. Yes, they are smart and honestly terrifying sometimes, but they are just **tools** at the end of the day.

Without a well versed user, tools are nothing more than scrap. This is where programmers come in. Until the day that machines are truly intelligent, we as programmers are the one who can use them effectively.

Programming is simply the language that translates human intention to computer actions. It instructs machines on how to carry out your logic.

So why bother learning how to code? So we can use cool tools like chatGPT or DALLE-2 to do amazing things!

Basic File Structure

Knowing how to structure files properly will save a lot of headaches in the future

Basic File Structure

Knowing how to structure files properly will save a lot of headaches in the future

1. Good File Names - avoid using confusing naming conventions

- use '_' instead of ' ' so that files are easier to navigate
- name things clearly and consistently (ex. homework_1, homework_2 instead of hw_1, homework_2)

Basic File Structure

Knowing how to structure files properly will save a lot of headaches in the future

1. Good File Names - avoid using confusing naming conventions

- use '_' instead of ' ' so that files are easier to navigate
- name things clearly and consistently (ex. homework_1, homework_2 instead of hw_1, homework_2)

2. Use Folders Where Possible - containing files

- in the future, assignments may come with accompanying files to work with
- storing all the files related to homework 1 in a folder can save a lot of pain
- ex. instead of having homework_1.py and files/ in your homework folder, have them in a folder named homework_1 that's stored in your homework folder

Basic File Structure

Knowing how to structure files properly will save a lot of headaches in the future

1. Good File Names - avoid using confusing naming conventions

- use '_' instead of ' ' so that files are easier to navigate
- name things clearly and consistently (ex. homework_1, homework_2 instead of hw_1, homework_2)

2. Use Folders Where Possible - containing files

- in the future, assignments may come with accompanying files to work with
- storing all the files related to homework 1 in a folder can save a lot of pain
- ex. instead of having homework_1.py and files/ in your homework folder, have them in a folder named homework_1 that's stored in your homework folder

Yes, doing so will mean more folders to navigate. But trust me, it will be a good practice for future classes and work

Loop Overview

Loops are used to **repeat computations** many times.

- Two types of loops:
    - While: Uses logical conditions, do not know the number of iterations (as long as a condition is true, code will run)
    - For loop: Usually for when the number repetition is known.

Loops are used to **repeat computations** many times.

- Two types of loops:
    - While: Uses logical conditions, do not know the number of iterations (as long as a condition is true, code will run)
    - For loop: Usually for when the number repetition is known.

- loops are used in a lot of existing code, they are very versatile and highly applicable
- they are a cornerstone concept of coding, make sure to get very familiar with them

Syntax

for (element) **in** (iterable):

    // do something

element: a variable name to track the item (common practice to use 'i' as the name)

iterable: some item that can be traversed through (list, dictionary, etc.)

while (condition):

    // do something

condition: an 'if' statement of item that evaluates to True/False that dictates when the loop ends

Style - Variables

It definitely matters!!

Good Variable Names

- Try to avoid using single letter names or built-in function names
- Avoid using the written out form of numbers as the variable name
    - ex: two = 2
    - This is undescriptive and can be misleading if the value 2 is changed to a different number in the future
    - ex: len = 4
    - This uses a built-in function's name which will overwrite the function (len will no longer work)
- Write code in a way such that if you from a year in the future came back to take a look, you wouldn't need to spend 20 minutes understanding your logic
- Additional style info can be found in this style guide. Make sure to follow all requirements!

Style Example

- Write a function that takes in a string and performs the following:
    - When you encounter a vowel ('a', 'e', 'i', 'o', 'u'), make it uppercase
    - When you encounter a number, increase it by 3
    - Keep all other characters as is
    - ex: "abc1" → "Abc4"

Note: You don't have to worry about understanding this code at the moment ~ this is just an example to show the difference between good and bad style!

This is bad style:

```
def example(string):
    """
    Function that alters a string by making all vowels uppercase, adds 2 to numbers, and skips punctuation, special characters, and spaces
    """
    three = 3
    areVOWELS = ['a', 'e', 'i', 'o', 'u']
    a = ''

    for char in string:
        if char in areVOWELS: #checks if the character is a vowel
            a += char.upper()
        elif char.isdigit(): #checks if the character is a number
            b = int(char) + 3
            a += str(b)
        else:
            a += char
```

This is good style:

```python
def example(string):
    """
    Function that alters a string by making all vowels uppercase, adds 2 to
    numbers, and skips punctuation, special characters, and spaces
    """
    num_increase = 3
    are_vowels = ['a', 'e', 'i', 'o', 'u']
    altered_str = ''

    for char in string:
        if char in are_vowels: #checks if the character is a vowel
            altered_str += char.upper()
        elif char.isdigit(): #checks if the character is a number
            updated_num = int(char) + num_increase
            altered_str += str(updated_num)
        else:
            altered_str += char
```

Practice Activity

Live coding + python tutor demo

1. Write a 'for' loop that prints the numbers from 1 to 10

Solution:

```
for i in range(10):
    print(i + 1)
```

2. Write a while loop that counts down from 10,000 in increments of 7 until a non-positive number is reached, printing the value of every operation before the loop ends

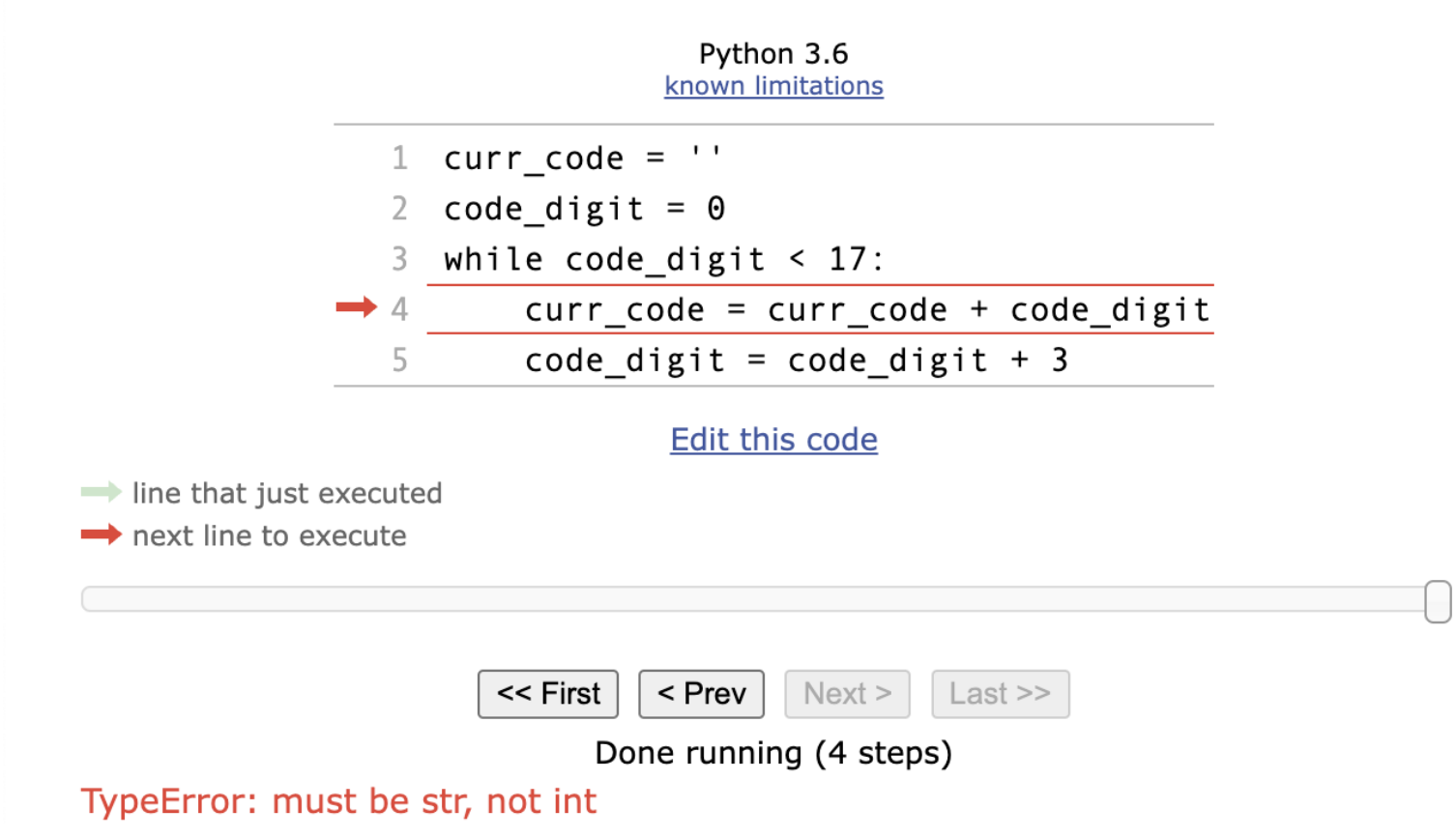
Solution:

```
curr_num = 10000
while curr_num > 0:
    print(curr_num)
    curr_num = curr_num - 7
```

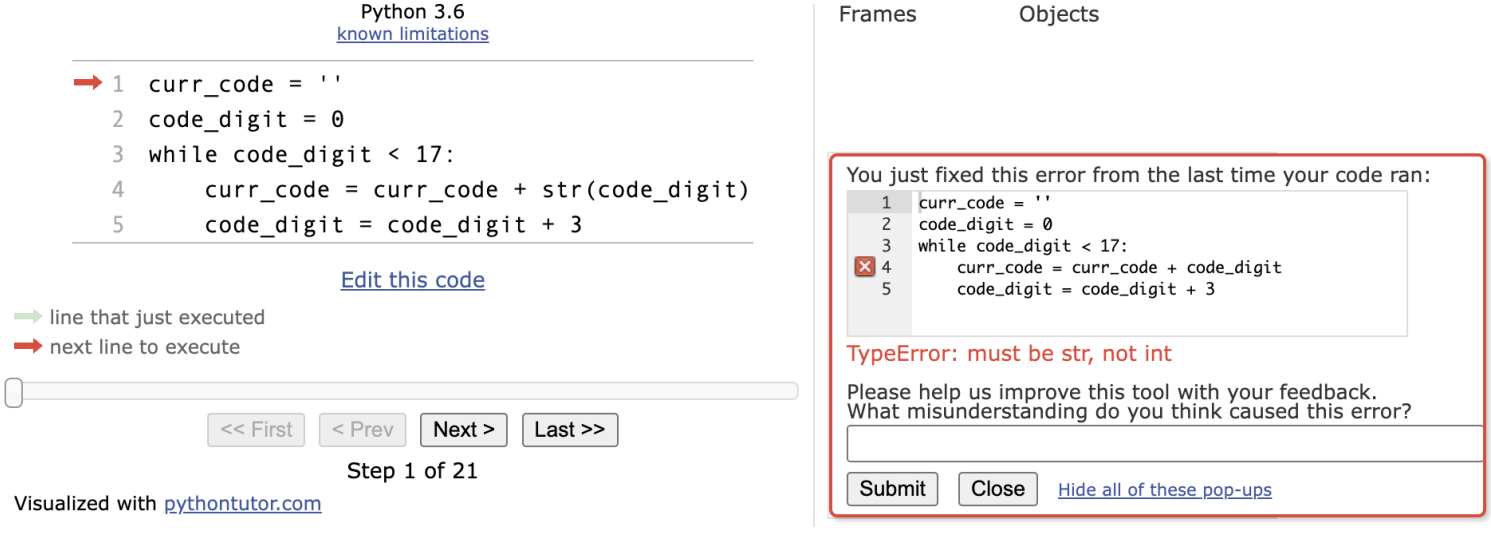
3. Write a while loop that generates a numeric code that consists of multiples of 3 that are less than 17.

Expected output: '03691215'

Erroneous Solution:



Corrected Solution:



Class Advice

Things I would do if I were a student taking DSC20 now

1. Utilize Resources
2. Use Online Content, don't depend on them
3. Take a break!
4. Useful Tools
5. Practice

1. Utilize Resources

- Office hours are amazing, not just for getting help on assignments or studying for the class, but as interpersonal resources. All the tutors are amazing and they're open to just talking as peers - they're people you can ask about future classes, career advice, etc. Some tutors will have more input than others, but don't be shy! We're students too

- We plan on making discussions particularly helpful this quarter. Rather than just reinforcing class concepts, we plan to also teach side topics that will help you become a strong programmer and help you adjust to exams (paper coding). Attendance is **not mandatory**, but strongly encouraged for newer programmers

- If you're ever struggling, don't be afraid to reach out to the staff for help. We do our best to support students

2. Use Online Content, don’t depend on them

- Tools like ChatGPT are amazing, but abusing them will only hurt your own learning
- Copying code from medium articles or w3school can get you a quick grade, but teaches you nothing and can get you AI'd
- Asking friends for a question can get not only you AI'd, but also your friend
- This class is best traversed with class content and staff support. You can consult your peers on programming ideas, but not the implementation itself

3. Take a break!

- parking at the desk for 3+ hours and staring at a screen never helped anyone
- remember to take breaks while coding, sometimes your brain just comes up with the solution
- I personally have a lot of solutions come to me after a shower

4. Useful Tools

1. Python Tutor - Debugging
    - visualizes code step by step that you enter
    - invaluable for debugging, amazing for understanding why some code works, and why others don't
2. w3school, geeksforgeeks, other documentation and example websites - Reference
    - Stores documentation for what functions/methods do with examples
    - Always includes examples so that you can understand what's actually happening
    - useful for quick refresher on what something does, amazing for figuring out new methods
3. Fig - File Navigation
    - terminal plugin for mac that autocompletes file names
    - can help you navigate your files much more easily

5. Practice

- like all new skills, practice makes perfect
- staring at code and not writing any will ultimately do nothing to make you a better coder
- Websites such as coding bat has practice problems for you to familiarize yourself with python
- codecademy is something that I personally used with DSC20 to help learn coding. It walks you through the fundamental functions of python and always frames it in sample code
- We also have a set of questions that you can practice off of, solutions can be posted next discussion.

Thanks for coming!