Teaching Computer Science College Courses in Prison

NCHEP 2023

Emma Hogan

Introduction

- Research
- Course details
- Motivation

Introduction

- Ph.D. student in Computer Science (CS) at UCSD
- Computing Education Research Lab
  - Studying effective pedagogies to help diverse students succeed in computing courses
- Dissertation topic: Improving computing education for incarcerated college students
  - Funding: NSF Broadening Participation in Computing (BPC)

Course Details

- Introduction to Python Programming
  - Currently teaching second iteration (Fall 2023 & 2022)
  - Fulfills requirement for students pursuing B.A. in Sociology
- Medium-maximum security facility
  - Many students lack exposure to modern technology
  - Laptops for education were newly debuted
    1. No code interpreter (students were unable to run code)

Motivation: Benefits of CS Education in Prisons

1. Incarcerated students and their families benefit:
    - In-demand job skills for post-release
    - Diversity of programs offered through HEP
2. Computing industry benefits:
    - Better tools and innovations through the inclusion of more diverse perspectives

<u>Demonstration</u>

The “Debugging” Process

The “Debugging” Process

“Debugging” in Computer Programming

- The “Debugging” Process:
  - Run code
  - Decipher error messages or unexpected result
  - Adjust based on any errors
- Critical for learning to program

“Debugging” in Computer Programming

- The “Debugging” Process:
  - Run code
  - Decipher error messages or unexpected result
  - Adjust based on any errors
- Critical for learning to program

Problem: How can we simulate the debugging process for students without a code interpreter?

Adaptations for Prison Environment

- Technology infrastructure
- Strategies without computer
- Using LMS features

Technology Infrastructure

Individual student laptops:

- Modified version of Canvas LMS
  - Otherwise no access to internet resources
- Microsoft Office (Word, Excel, Powerpoint)
- No code interpreter

In first iteration (Fall 2022) chose **not** to rely on Canvas:

- Many students were still learning basic use

Adaptations for No Code Interpreter

- Opportunity to stress conceptual understanding

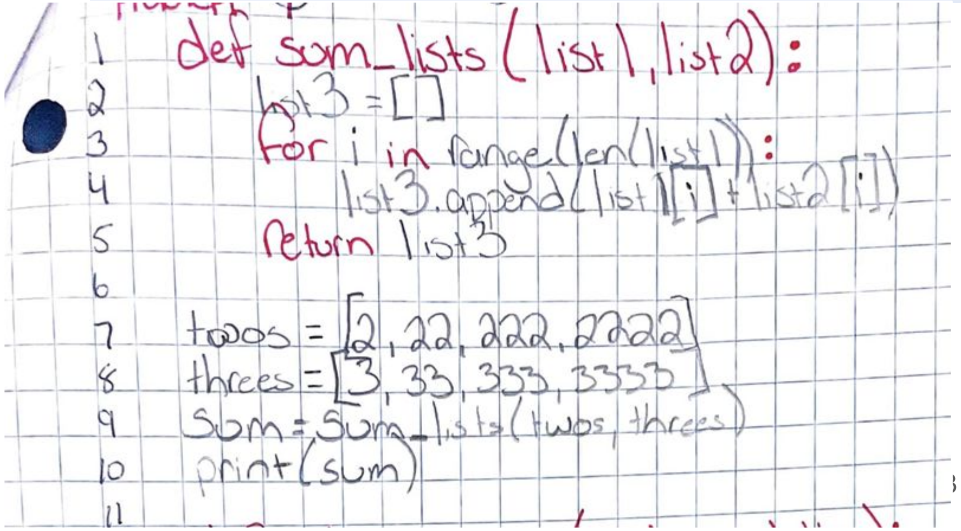
→

- Recent work reveals “hack-until-it-works” mentality is damaging to novice learning in CS
- Split homework grade between PAs and “Problem Sets” with paper-based activities (i.e., tracing, short answer, fill-in-the-blank)

Adaptations for No Code Interpreter

- Opportunity to stress conceptual understanding
- Handwriting code

Instructor transcribed & printed output 2-3 times per week



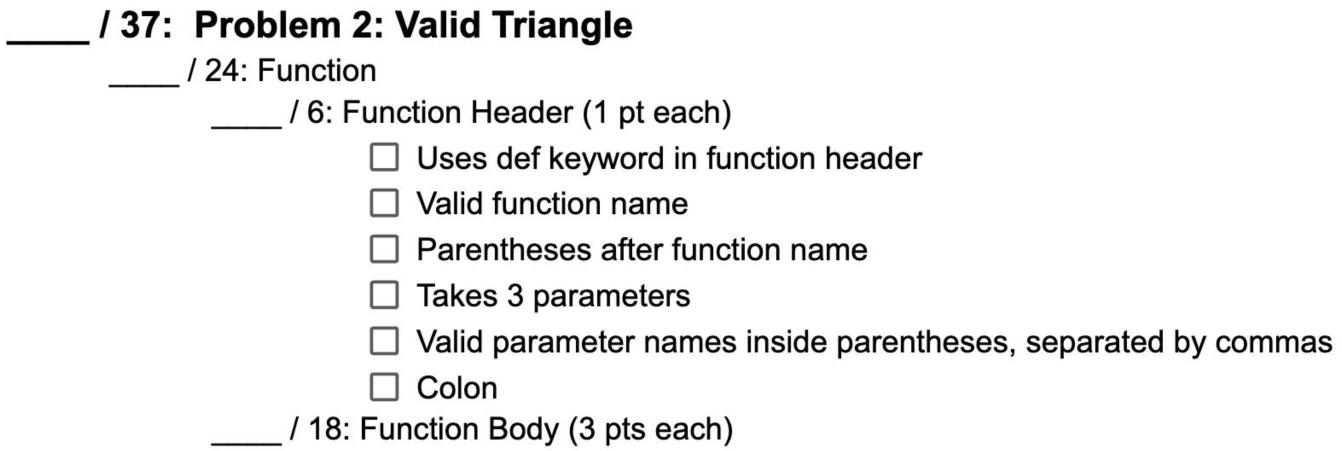
```
def sum_lists(list1, list2):
    list3 = []
    for i in range(len(list1)):
        list3.append(list1[i] + list2[i])
    return list3

twos = [2, 22, 222, 2222]
threes = [3, 33, 333, 3333]
sum = sum_lists(twos, threes)
print(sum)
```

Adaptations for No Code Interpreter

- Opportunity to stress conceptual understanding
- Handwriting code
- Detailed Grading Rubrics

Partial credit & detailed feedback especially important without code interpreter



____ / 37: Problem 2: Valid Triangle

____ / 24: Function

____ / 6: Function Header (1 pt each)

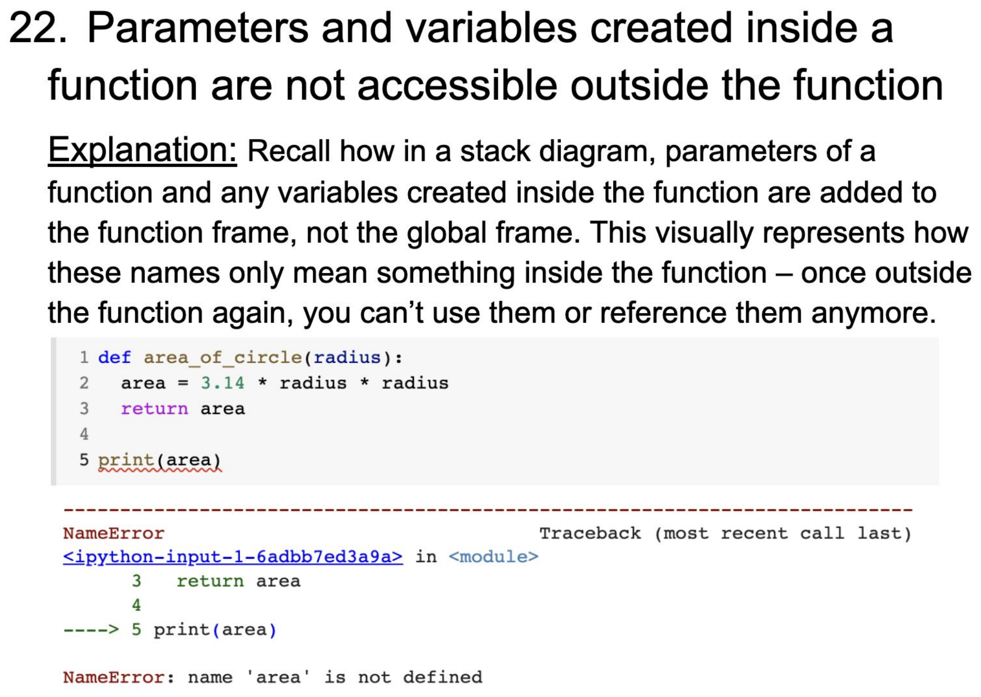
- ☐ Uses def keyword in function header
- ☐ Valid function name
- ☐ Parentheses after function name
- ☐ Takes 3 parameters
- ☐ Valid parameter names inside parentheses, separated by commas
- ☐ Colon

____ / 18: Function Body (3 pts each)

Adaptations for No Code Interpreter

- Opportunity to stress conceptual understanding
- Handwriting code
- Detailed Grading Rubrics
- Common Errors Resource

Running list of common errors with explanation, example, & error message



22. Parameters and variables created inside a function are not accessible outside the function

Explanation: Recall how in a stack diagram, parameters of a function and any variables created inside the function are added to the function frame, not the global frame. This visually represents how these names only mean something inside the function – once outside the function again, you can't use them or reference them anymore.

```
1 def area_of_circle(radius):
2   area = 3.14 * radius * radius
3   return area
4
5 print(area)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-6adbb7ed3a9a> in <module>
      3   return area
      4
----> 5 print(area)

NameError: name 'area' is not defined
```

Using LMS Features

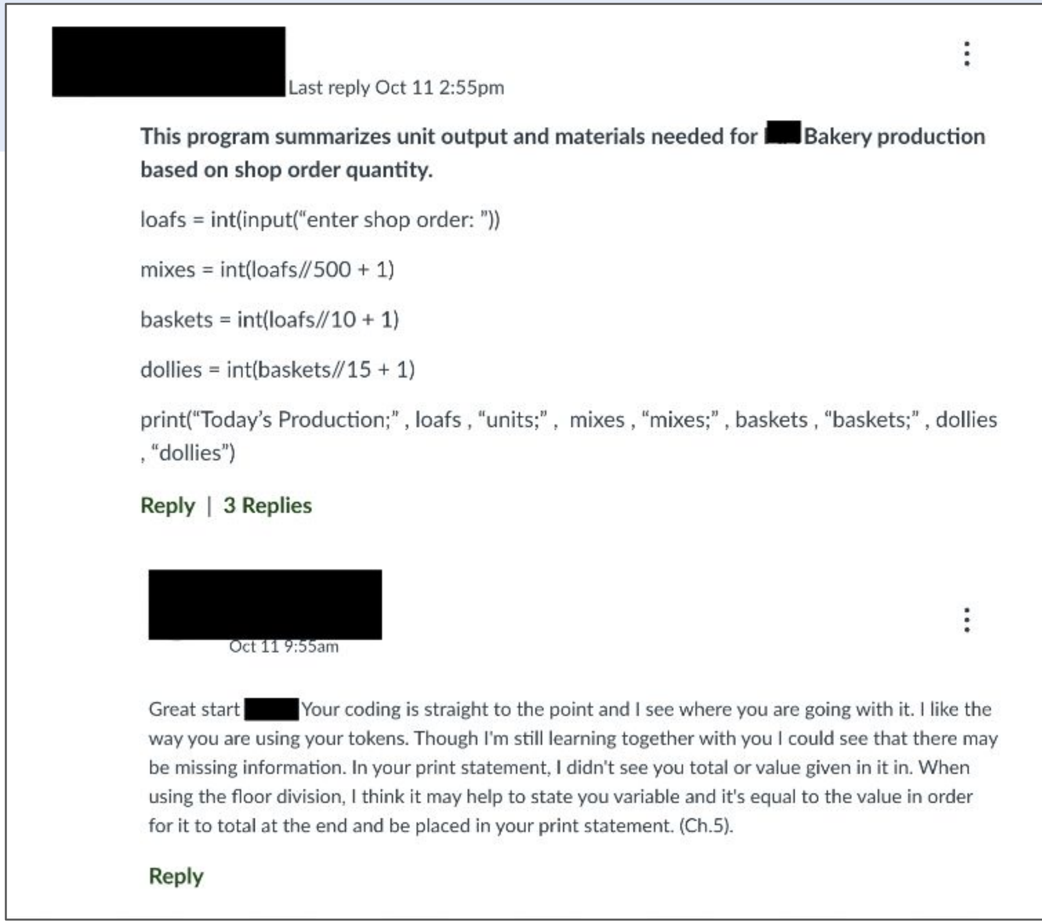
- Daily Code Runs

  - Students submit code draft in Canvas quiz
  - Code is run, output and feedback returned to student in submission comments section

Using LMS Features

- Daily Code Runs
- Code Reviews

- Peer feedback midway through assignment period
- Students post draft of individual code
- Give feedback to at least 3 peers



[redacted] Last reply Oct 11 2:55pm

This program summarizes unit output and materials needed for [redacted] Bakery production based on shop order quantity.

```
loafs = int(input("enter shop order: "))

mixes = int(loafs//500 + 1)

baskets = int(loafs//10 + 1)

dollies = int(baskets//15 + 1)

print("Today's Production;" , loafs , "units;" ,  mixes , "mixes;" , baskets , "baskets;" , dollies , "dollies")
```

Reply | 3 Replies

[redacted] Oct 11 9:55am

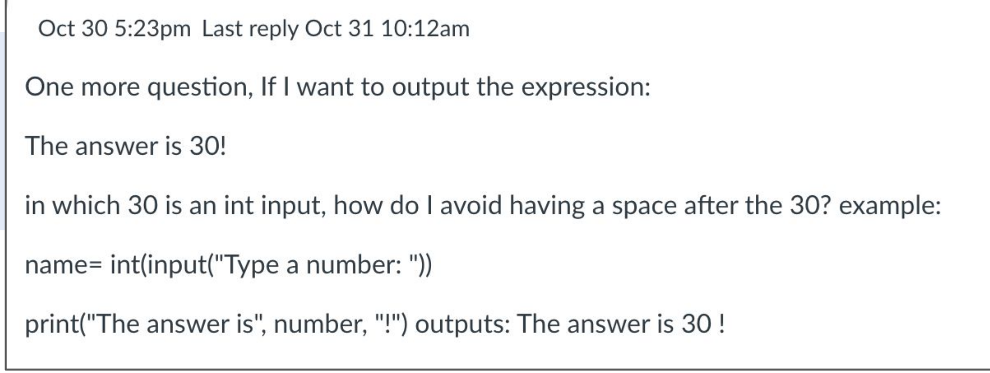
Great start [redacted] Your coding is straight to the point and I see where you are going with it. I like the way you are using your tokens. Though I'm still learning together with you I could see that there may be missing information. In your print statement, I didn't see you total or value given in it in. When using the floor division, I think it may help to state you variable and it's equal to the value in order for it to total at the end and be placed in your print statement. (Ch.5).

Reply

Using LMS Features

- Daily Code Runs
- Code Reviews
- Discussion Board: Open Q&A

- Communication w/ students
- In-depth explanations (including media option)
- Questions outside the scope of course



Oct 30 5:23pm Last reply Oct 31 10:12am

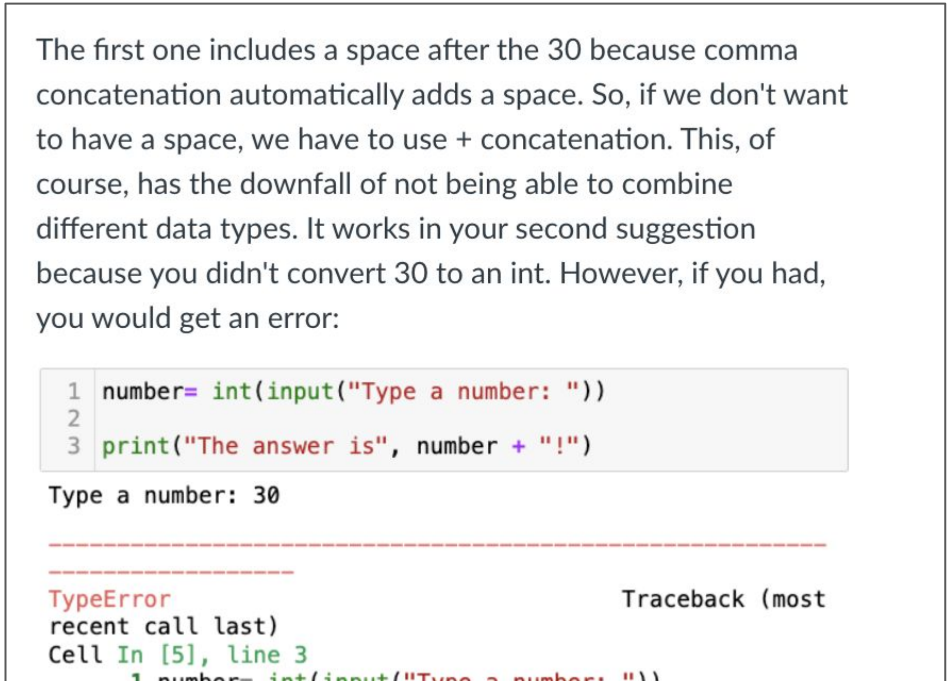
One more question, If I want to output the expression:

The answer is 30!

in which 30 is an int input, how do I avoid having a space after the 30? example:

name= int(input("Type a number: "))

print("The answer is", number, "!") outputs: The answer is 30 !



The first one includes a space after the 30 because comma concatenation automatically adds a space. So, if we don't want to have a space, we have to use + concatenation. This, of course, has the downfall of not being able to combine different data types. It works in your second suggestion because you didn't convert 30 to an int. However, if you had, you would get an error:

```
number= int(input("Type a number: "))

print("The answer is", number + "!")
```

```
Type a number: 30

---------------------------------------------------------------------------
TypeError                                 Traceback (most
recent call last)
Cell In [5], line 3
      1 number= int(input("Type a number: "))
```

Conclusion: Future Plans

Future Work

- Continue iterative improvements to introductory CS course in future offerings
- Expand variety of CS courses offered
- Expand to more prison settings

Thank you!

Contact: Emma Hogan; UC San Diego

- emhogan@ucsd.edu
- linkedin.com/in/emmamhogan/

Acknowledgments: Thank you to my research collaborators, Ruoxuan Li and advisors Leo Porter, Bill Griswold, and Gerald Soosai Raj!