Response from Wednesday's "how are you feeling" poll

Everything is nice and positive with the help of the awesome TAs of cs61A they always know how to hype everyone up and keep cool under pressure

Midterm 1 Studying Advice

pollev.com/cs61a

Amy

Carolyn

Apollo

Esha

Rajoshi

student support: Ask about exam prep guidance or any other advising in 1:1 context

Midterm Review

Announcements

Fall 2022 Midterm 1 Question 1(b)

```
bear = -1
oski = lambda print: print(bear)
bear = -2
print(oski(abs))
```

pollev.com/cs61a

https://pythontutor.com/cp/composingprograms.html#code=bear%20%3D%20-1%0Aoski%20%3D%20lambda%20print%3A%20print%28bear%29%0Abear%20%3D%20-2%0Aprint%28oski%28abs%29%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Conditional Expressions Practice

Fall 2022 Midterm 1 Question 1

```
(3 and 4) - 5
```

and, **or**: Always give you the value of either the expression on the left or the expression on the right

pollev.com/cs61a

True and False Values

The built-in bool(x) returns True for true x and False for false x.

```
>>> bool(0)
False
>>> bool(-1)
True
>>> bool(0.0)
False
>>> bool(' ')
True
>>> bool('')
False
>>> bool(False)
False
>>> bool(print('fool'))
fool
False
```

Environment Diagram Practice

Assigning Names to Values

There are three ways of assigning a name to a value:

- Assignment statements (e.g., y = x) assign names in the current frame
- Def statements assign names in the current frame
- Call expressions assign arguments new names in a local frame

```
h = lambda f: lambda x: f(f(x))
h(abs)(-3)
```

```
f = abs
x = -3
f(f(x))
```

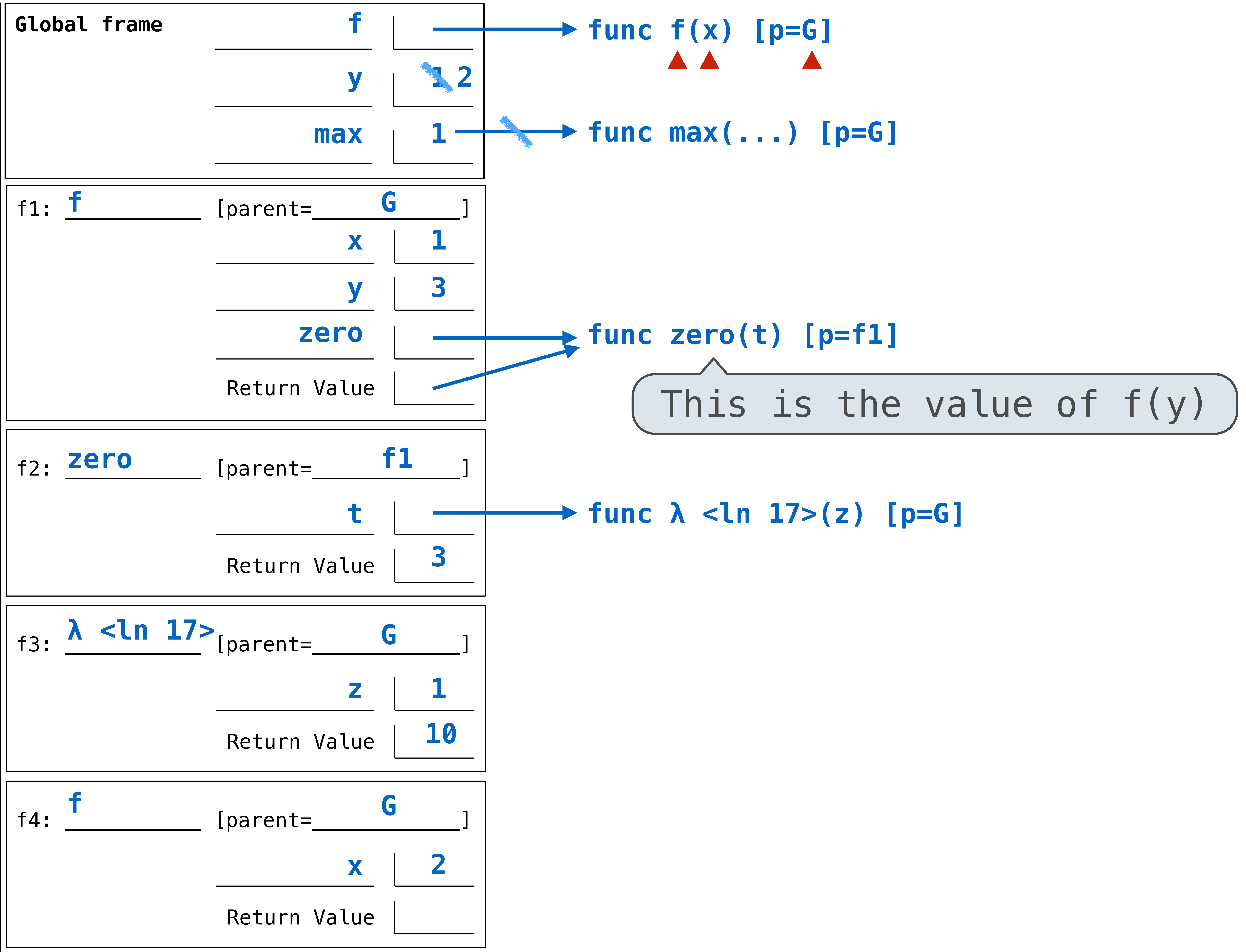
Fall 2022 CS 61A Midterm 1, Question 2

When a function is called:

1. Add a **local frame**, titled with the *<name>* of the function being called.
2. Copy the parent of the function to the **local frame**: [parent=*<label>*]
3. Bind the *<formal parameters>* to the arguments in the **local frame**.
4. Execute the body of the function in the environment that starts with the **local frame**.

- **The Diagram**
- **Annotations**

```
1: def f(x):
2:     """f(x)(t) returns max(x*x, 3*x)
3:     if t(x) > 0, and 0 otherwise.
4:     """
5:     y = max(x * x, 3 * x)
6:     def zero(t):
7:         if t(x) > 0:
8:             return y
9:         return 0
10:    return zero
11:
12: # Find the largest positive y below 10
13: # for which f(y)(lambda z: z - y + 10)
14: # is not 0.
15: y = 1
16: while y < 10:
17:     if f(y)(lambda z: z - y + 10):
18:         max = y
19:     y = y + 1
```



https://pythontutor.com/cp/composingprograms.html#code=def%20f%28x%29%3A%0A%20%20%20%20%22%22%22f%28x%29%28t%29%20returns%20max%28x*x,%203*x%29%20%0A%20%20%20%20if%20t%28x%29%20%3E%200,%20and%200%20otherwise.%0A%20%20%20%20%22%22%22%0A%20%20%20%20y%20%3D%20max%28x%20*%20x,
%203%20*%20x%29%0A%20%20%20%20def%20zero%28t%29%3A%20%0A%20%20%20%20%20%20%20%20if%20t%28x%29%20%3E%200%3A%0A%20%20%20%20%20%20%20%20%20%20%20%20return%20y%0A%20%20%20%20%20%20%20%20return%200%0A%20%20%20%20return%20zero%0A%0A%23%20Find%20the%20largest%20positive%20y%20below%2010%0A%23%20for%20which%20f%28y%29%28lambda%20z%3A%20z%20-
%20y%20%2B%2010%29%0A%23%20is%20not%200.%0Ay%20%3D%201%0Awhile%20y%20%3C%2010%3A%20%20%20%20%0A%20%20%20%20if%20f%28y%29%28lambda%20z%3A%20z%20-%20y%20%2B%2010%29%3A%0A%20%20%20%20%20%20%20%20max%20%3D%20y%0A%20%20%20%20y%20%3D%20y%20%2B%201%0A%0A&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Function Implementation Practice

A Slight Variant of Fall 2022 Midterm 1 3(b)

Implement nearest_prime, which takes an integer n above 5. It returns the nearest prime number to n. If two prime numbers are equally close to n, return the larger one. Assume is_prime(n) is implemented already.

Example: n is 21

```
def nearest_prime(n):
    """Return the nearest prime number to n.
    In a tie, return the larger one.

    >>> nearest_prime(8)
    7
    >>> nearest_prime(11)
    11
    >>> nearest_prime(21)
    23
    """
    k = 0
    while True:
        if _is_prime(23)_:
            return __23__
        if k > 0:
            k = -k
        else:
            k = ________
```

keep looking for a prime

From discussion:

Describe a process (in English) that computes the output from the input using simple steps.

Figure out what additional names you'll need to carry out this process.

Implement the process in code using those additional names.

Read the description

Verify the examples & pick a simple one

Read the template

Annotate names with values from your chosen example

Write code to compute the result

Did you really return the right thing?

Check your solution with the other examples

A Slight Variant of Fall 2022 Midterm 1 3(b)

Implement nearest_prime, which takes an integer n above 5. It returns the nearest prime number to n. If two prime numbers are equally close to n, return the larger one. Assume is_prime(n) is implemented already.

```
def nearest_prime(n):     Example: n is 21
    """Return the nearest prime number to n.
    In a tie, return the larger one.

    >>> nearest_prime(8)
    7
    >>> nearest_prime(11)
    11
    >>> nearest_prime(21)
    23
    """
    k = 0
    while True:
        if is_prime(n + k): is_prime(23)
            return  n + k   23
        if k > 0:
            k = -k
        else:
            k = -k + 1
```

keep looking for a prime

From discussion:

Describe a process (in English) that computes the output from the input using simple steps.

Figure out what additional names you'll need to carry out this process.

Implement the process in code using those additional names.

Process:
Check whether a number is prime in this order:
- original n
- n + 1
- n − 1
- n + 2
- n − 2
- n + 3
- n − 3
- n + 4
...

All of these look like n + k for various k