Function Examples

Announcements

Lab Review: Call Expressions

Lab 02 Q2: Higher-Order Functions

```
>>> def cake():
...     print('beets')
...     def pie():
...         print('sweets')
...         return 'cake'
...     return pie
...
>>> chocolate = cake()
beets
>>> chocolate
<function cake.<locals>.pie at ...>
>>> chocolate()
sweets
'cake'
```

```
>>> def snake(x, y):
...     if cake == more_cake:
...         return chocolate
...     else:
...         return x + y
...
>>> snake(10, 20)
<function cake.<locals>.pie at ...>
>>> snake(10, 20)()
sweets
'cake'
>>> cake = 'cake'
>>> snake(10, 20)
30
```

```
>>> more_chocolate, more_cake = chocolate(), cake
sweets
>>> more_chocolate
'cake'
```

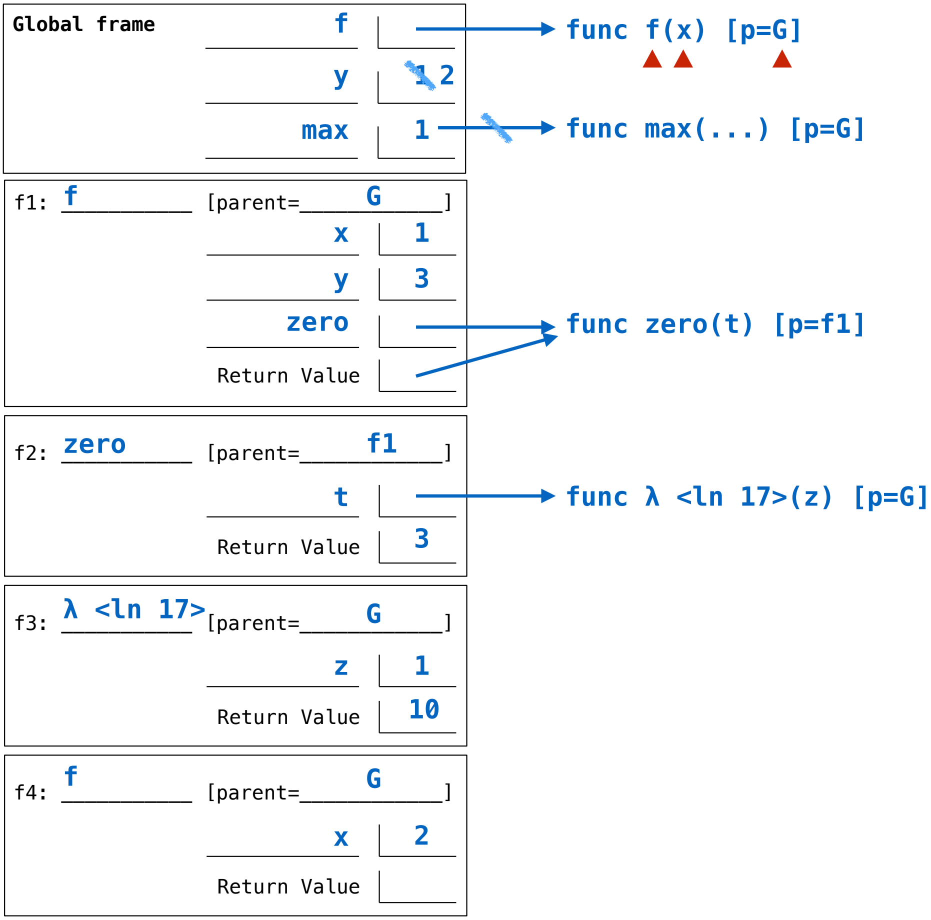
https://pythontutor.com/cp/composingprograms.html#code=def%20cake%28%29%3A%0A%20%20%20print%28'beets'%29%0A%20%20%20def%20pie%28%29%3A%0A%20%20%20%20%20%20%20print%28'sweets'%29%0A%20%20%20%20%20%20%20return%20'cake'%0A%20%20%20return%20pie%0A%0Achocolate%20%3D%20cake%28%29%0A%23%20chocolate%0A%23%20chocolate%28%29%0Amore_chocolate,
%20more_cake%20%3D%20chocolate%28%29,%20cake%0A%23%20more_chocolate%0Adef%20snake%28x,
%20y%29%3A%0A%20%20%20if%20cake%20%3D%3D%20more_cake%3A%0A%20%20%20%20%20%20%20return%20chocolate%0A%20%20%20else%3A%0A%20%20%20%20%20%20%20return%20x%20%2B%20y%0A%0A%23%20snake%2810,%2020%29%0Asnake%2810,%2020%29%28%29%0A%23%20cake%20%3D%20'cake'%0A%23%20snake%2810,%2020%29&cumulative=true&curInstr=0&mode=display&origin=composingprograms.js&py=3&rawInputLstJSON=%5B%5D

Environment Diagram Practice

- The Diagram
- Annotations

Fall 2022 CS 61A Midterm 1, Question 2

```
 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.

```
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(23)_:
            return ___23___
        if k > 0:              keep
            k = -k             looking
        else:                  for a
            k = ________       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

(Demo)

Currying

Function Currying

```
def make_adder(n):
    return lambda k: n + k
```

```
>>> make_adder(2)(3)
5
>>> add(2, 3)
5
```

There's a general relationship between these functions

(Demo)

Curry: Transform a multi-argument function into a single-argument, higher-order function

Example: Reverse

The square function can be defined in terms of the built-in pow function:

```
def square(x):
    """Square x.

    >>> square(3)
    9
    """
    return pow(x, 2)
```

```
def cube(x):
    """Cube x.

    >>> cube(3)
    27
    """
    return pow(x, 3)
```

Define square and cube in one line without using lambda or ** (using curry and reverse).

```
def reverse(f):
    return lambda x, y: f(y, x)
```

```
def curry(f):
    def g(x):
        def h(y):
            return f(x, y)
        return h
    return g
```

```
square = curry(reverse(pow))(2)
cube   = curry(reverse(pow))(3)
```