

Q2: (Tutorial) Warm Up: Case Conundrum

These exercises are meant to help refresh your memory of topics covered in lecture and/or lab this week before tackling more challenging problems.

In this question, we will explore the difference between the `if` and `elif` keywords.

What is the result of evaluating the following 3 pieces of code? Each column is a separate problem.

```
def special_case():  
    x = 10  
    if x > 0:  
        x += 2  
    elif x < 13:  
        x += 3  
    elif x % 2 == 1:  
        x += 4  
    return x
```

`special_case()`

12

```
def just_in_case():  
    x = 10  
    if x > 0:  
        x += 2  
    if x < 13:  
        x += 3  
    if x % 2 == 1:  
        x += 4  
    return x
```

`just_in_case()`

14

```
def case_in_point():  
    x = 10  
    if x > 0:  
        return x + 2  
    if x < 13:  
        return x + 3  
    if x % 2 == 1:  
        return x + 4  
    return x
```

`case_in_point()`

12



Which of these code snippets result in the same output, and why? Based on your findings, when do you think using a series of `if` statements has the same effect as using both `if` and `elif` cases?

Q4: (Tutorial) Is Prime?

Write a function that returns `True` if a positive integer `n` is a prime number and `False` otherwise.

A prime number `n` is a number that is not divisible by any numbers other than 1 and `n` itself. For example, 13 is prime, since it is only divisible by 1 and 13, but 14 is not, since it is divisible by 1, 2, 7, and 14.

```
def is_prime(n):  
    """  
    >>> is_prime(10)  
    False  
    >>> is_prime(7)  
    True  
    """  
    """ YOUR CODE HERE """  
    if n == 1:  
        return False  
  
    else:  
        x = 2  
        while x < n:  
            if n % x == 0:  
                return False  
  
            x += 1  
  
        return True
```

→ 1 is not prime

- while loop to go from 2... N
- check remainder of input / current number
 - if 0, return False
- if it goes through everything, return True

Hint: Use the `%` operator: `x % y` returns the remainder of `x` when divided by `y`.

Q5: (Tutorial) Fizzbuzz

Implement `fizzbuzz(n)`, which prints numbers from 1 to `n`. However, for numbers divisible by 3, print "fizz". For numbers divisible by 5, print "buzz". For numbers divisible by both 3 and 5, print "fizzbuzz".

This is a standard software engineering interview question, but we're confident in your ability to solve it!

```
def fizzbuzz(n):  
    """  
  
    >>> result = fizzbuzz(16)  
    1  
    2  
    fizz  
    4  
    buzz  
    fizz  
    7  
    8  
    fizz  
    buzz  
    11  
    fizz  
    13  
    14  
    fizzbuzz  
    16  
    >>> result == None  
    True  
    """  
  
    "*** YOUR CODE HERE ***"
```

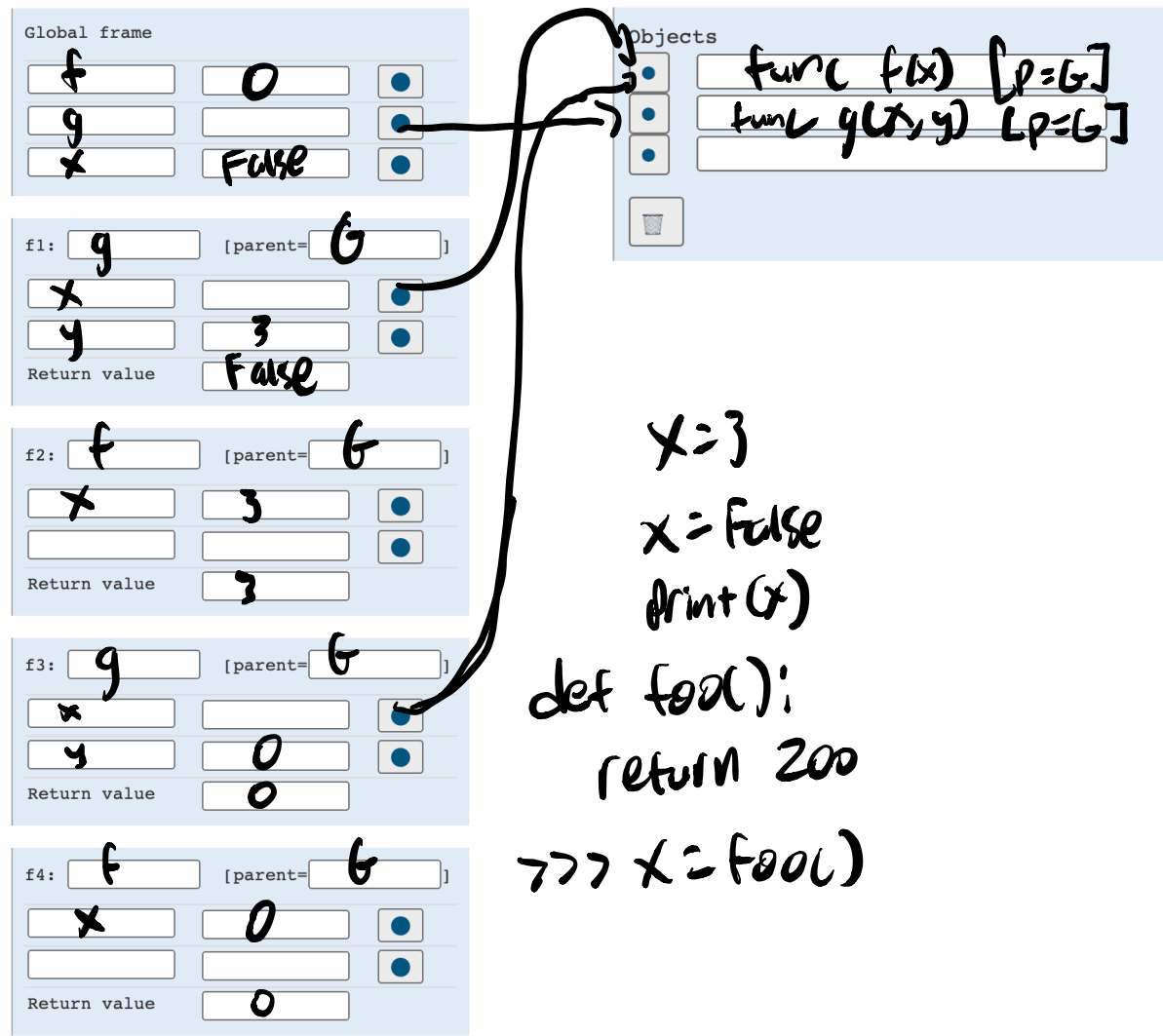
Q9: (Tutorial) Nested Calls Diagrams

Draw the environment diagram that results from executing the code below. You may not need to use all of the frames and blanks provided to you.

```
def f(x):  
    return x  
  
def g(x, y):  
    if x(y):  
        return not y  
    return y  
  
x = 3  
x = g(f, x)  
f = g(f, 0)
```

Evaluation Procedure

- 1. Evaluate Operator
- 2. Evaluate operands
- 3. Apply operator
 - operands
 - in step 3, open up a new frame!



def foo():

def foo2():