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.
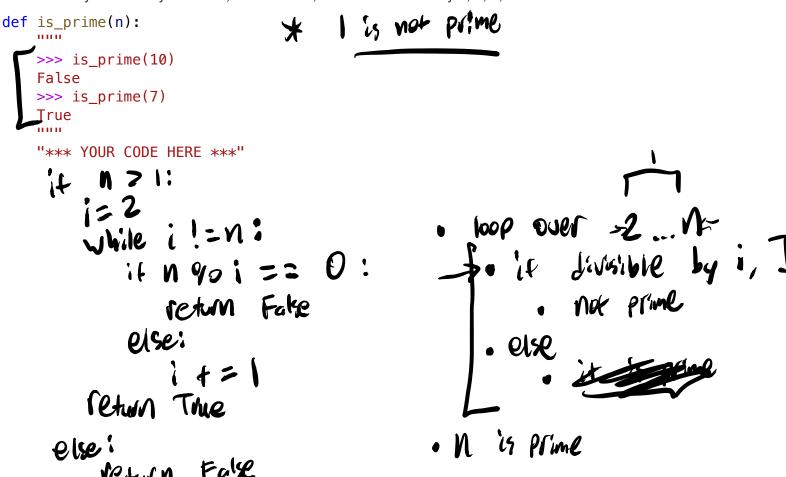
```python
def special_case():
    x = 10
    if x > 0:
        x += 2      x=12  ✓
    elif x < 13:
        x += 3
    elif x % 2 == 1:
        x += 4
    return x

special_case()
```
12

```python
def just_in_case():
    x = 10
    if x > 0:
        x += 2   x=12
    if x < 13:
        x += 3   x=15
    if x % 2 == 1:
        x += 4
        x=19
    return x

just_in_case()
```
19

```python
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.

```python
def is_prime(n):
    """
    >>> is_prime(10)
    False
    >>> is_prime(7)
    True
    """
    "*** YOUR CODE HERE ***"
```

* 1 is not prime

```
if n > 1:
    i = 2
    while i != n:
        if n % i == 0:
            return False
        else:
            i += 1
    return True
else:
    return False
```

- loop over 2 ... n
  → • if divisible by i,]
    • not prime
  • else
    • ~~it is prime~~
- n is prime

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!

```python
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!

Global frame → value

name	value	
f	O	●
g		●
x	False	●

Objects

func f(x) [p=G]
func g(x,y) [p=G]

f1: g   [parent= G]

x	⇒	
y	⇒	3

Return value   False

f2: f   [parent= G]

| x | 3 | ● |
| | | ● |

Return value   3

f3: g   [parent= G]

| x | | ● |
| y | O | ● |

Return value   O

f4: f   [parent= G]

| x | O | ● |
| | | ● |

Return value   O