Simplifying If-Statements

Introduction to Computer Programming

Dr. Paul Vrbik

September 26, 2018

Definition (Factoring)

In computer science factoring is breaking a complex problem into parts that are easier to conceive, understand, program, and maintain.

Definition (Refactoring)

Code refactoring is the process of restructuring existing computer code – changing the factoring – without changing its external behaviour.

Simplifying If-Statements

```
>>> if x>1:
...     if y>2:
...         if z>3:
...             print("hello")
```

simplifies too...

```
>>> if x>1 and y>2 and z>3:
...     print("hello")
```

Simplifying If-Statements

```
>>> def foo(x):
...     if x > 0:
...         return True
...     else:
...         return False
```

simplifies too...

```
>>> def foo(x):
...     return x > 0
```

Simplifying If-Statements

```
>>> if x > 0:
...     y = True
... else:
...     y = False
```

simplifies too...

```
>>> y = x > 0
```

Simplifying If-Statements

```
>>> if x > y == True:
...     z = 1
```

simplifies too...

```
>>> if x > y:
...     z = 1
```

Question

Are the following pieces of code equivalent?

```
>>> #Block 1
>>> if x > 0:
...     print("A")
... elif x <= 0 and x % 2 == 0:
...     print("B")

>>> #Block 2
>>> if x > 0:
...     print("A")
... elif x % 2 == 0:
...     print("B")
```

Yes!

```
>>> if x > 0:
...     print("A")
... elif x <= 0 and x % 2 == 0:
...     print("B")
```

is equivalent to...

```
>>> if x > 0:
...     print("A")
... elif x % 2 == 0:
...     print("B")
```

because here it must be that `x <= 0`

Question

Are the following pieces of code equivalent?

```
>>> #Block 1
>>> if x > 0:
...     print("A")
... if x <= 0 and x % 2 == 0:
...     print("B")

>>> #Block 2
>>> if x > 0:
...     print("A")
... if x % 2 == 0:
...     print("B")
```

No!

```
>>> if x > 0:
...     print("A")
... if x <= 0 and x % 2 == 0:
...     print("B")
```

is not equivalent to...

```
>>> if x > 0:
...     print("A")
... if x % 2 == 0:   this if is independent of the one before it
...     print("B")
```

Let `x = 2`. The first `if` prints `A` and the second prints `A` and `B`.

Question (Final 2014 Q1.D)

What is the value of `z` after executing the following?

```
>>> (x, y, z) = (1, 2, 3)
>>> if x < y:
...     if y > 4:
...         z = 5
...     else:
...         z = 6
```

Answer

`z = 6`.

Question (Final 2016 Q2.A)

Provide the correct conditions for the following code:

```
def decide_admission(average, cutoff):
    """(int, int) -> str
    Returns student offer message.  Students graded five more
    than the cut-off receive scholarships.
    """
    if cond0:
        return "accept"
    elif cond1:
        return "accept with scholarship"
    elif cond2:
        return "reject"
```

Answer (Final 2016 Q2.A)

```
def decide_admission(average, cutoff):
    """(int, int) -> str
    Returns student offer message.  Students graded five more
    than the cut-off receive scholarships.
    """
    if average >= cutoff and average-cutoff < 5:
        return "accept"
    elif average-cutoff >= 5:
        return "accept with scholarship"
    elif True:
        return "reject"
```

Follow up: Refactor this code.

```
def decide_admission(average, cutoff):
    if average >= cutoff:
        if average-cutoff >= 5:
            return "accept with scholarship"
        return "accept"
    return "reject"
```

Question (Final 2015 Q1.F)

Evaluate `f(2)`, `f(13)`, `f(-8)`, and `f(10)` when:

```
def f(x):
    if x %2 != 0:
        if x**2 < 36:
            return ‘Pow’
        else:
            return x // 3
    else:
        if x <0 and abs(x) > 5:
            return False
        elif not x + 2 > 8:
            return x / 2
    return ‘Zonk’
```

Answer (Final 2015 Q1.F)

1. `f(2) = 1.0`,
2. `f(13) = 4`,
3. `f(-8) = False`, and
4. `f(10) = 'Zonk'`.

Question (Final 2015 Q2)

What is the value of `total` after this code runs?

```
(total, first, second) = (0, 4, 0)
if first > 5:
    if first == 4:
        first = 6
        total = total + 1
    else:
        total = toral + 2
else:
    if first > 5:
        total = total + 1
```

Answer (Final 2015 Q2)

```
total = 0.
```

Question

Refactor the code in `L08Q01.py`.

Answer

```
def foo(x, y):
    return 'A'
```

Next Time

1. Looping! (Over stings)