CSC110 Lecture 6: If Statements

Hisbaan Noorani

September 23, 2020

Contents

1 Ex 1: Practice with if statements

1. Consider the code below. Note that the line are numbers on the left margin.

```
1  def can_vote(age: int) -> str:
2      """Return a string indicating whether age is a legal voting age in Canada.
3
4      In Canada, you must be at least 18 years old to vote.
5      """
6      if age < 18:
7          return 'Too young to vote'
8      else:
9          return 'Allowed to vote'
```

   (a) In the table below, write down the sequence of line numbers in the fucntions body that will be executed for the given input.

Age	Lines executed
17	1, 6, 7
18	1, 6, 9
19	1, 6, 9

   (b) Suppose we have the following doctest examples for this function.

```
1  >>> can_vote(1)
2  'Too young to vote'
3  >>> can_vote(2)
4  'Too young to vote'
```

   Why is this not a good choice of doctest examples?
   It only tests one of the branches of the if statement.

2. Implement each of the following functions using an if statement.

```
def format_name(first: str, last: str) -> str:
    """Return the first and last names as a single string in the form: last, first.

    Mononymous persons (those with no last name) should have
    their name returned without a comma.

    >>> format_name('Cherilyn', 'Sarkisian')
    'Sarkisian, Cherilyn'
    >>> format_name('Cher', '')
    'Cher'
    """
    if last == '':
        return first
    else:
        return last + ', ' + first


def larger_sum(nums1: list, nums2: list) -> list:
    """Return the list with the larger sum.

    Assume that nums1 and nums2 are lists of floats.
    In the event of a tie, return nums1.

    >>> larger_sum([1.26, 2.01, 3.30], [3.00, 3.00, 3.00])
    [3.00, 3.00, 3.00]
    >>> larger_sum([2.00, 1.00], [1.00, 2.00])
    [2.00, 1.00]
    """
    if sum(nums1) >= sum(nums2):
        return nums1
    else:
        return nums2
```

2 Ex 2: Multiple brances

1. Implement each of the following functions, using elifs to create if statements with more than two branches.

```
   def porridge_satisfaction(temperature: float) -> str:
    """Return what a picky eater says when tasting porridge with the given temperature.

    Temperatures greater than 50.0 are too hot, temperatures less than 49.0 are too cold,
    and the temperatures in between are just right.

    >>> porridge_satisfaction(65.5)
    'This porridge is too hot! Ack!!'
    >>> porridge_satisfaction(30.0)
    'This porridge is too cold! Brrr..'
    >>> porridge_satisfaction(49.5)
    'This porridge is just right! Yum!!'
    """
    if temperature > 50.0:
        return 'This porridge is too hot! Ack!!'
```

```
    elif temperature < 49.0:
        return 'This porridge is too cold! Brr..'
    else:
        return 'This porridge is just right! Yum!!'


def rock_paper_scissors(player1: str, player2: str) -> str:
    """Return the winner of a game of rock, paper, scissors.

    The game is played with the following rules:
        1) 'rock' wins against 'scissors'
        2) 'scissors' wins against 'paper'
        3) 'paper' wins against 'rock'

    Ties are allowed.

    You may assume that the input strings are in {'rock', 'paper', 'scissors'}.

    >>> rock_paper_scissors('rock', 'scissors')
    'Player1 wins'
    >>> rock_paper_scissors('rock', 'paper')
    'Player2 wins'
    >>> rock_paper_scissors('rock', 'rock')
    'Tie!'
    """
    if player1 == 'rock' and player2 == 'rock':
        return 'Tie!'
    elif player1 == 'rock' and player2 == 'paper':
        return 'Player2 wins'
    elif player1 == 'rock' and player2 == 'scissors':
        return 'Player1 wins'
    elif player1 == 'paper' and player2 == 'rock':
        return 'Player1 wins'
    elif player1 == 'paper' and player2 == 'paper':
        return 'Tie!'
    elif player1 == 'paper' and player2 == 'scissors':
        return 'Player1 wins'
    elif player1 == 'scissors' and player2 == 'rock':
        return 'Player2 wins'
    elif player1 == 'scissors' and player2 == 'paper':
        return 'Player1 wins'
    elif player1 == 'scissors' and player2 == 'scissors':
        return 'Tie!'
    else:
        return 'Error!'
```

3 Ex 3: Simplifying if statements

1. Even though this lecture is all about if statements, often we can implement predicates (functions that return booleans) without using if statements at all! For each of the following functions, rewrite the function body so that it does not use an if statement.

```python
def is_odd(n: int) -> bool:
    """Return whether n is odd (not divisible by 2).

    if n % 2 == 0:
        return False
    else:
        return True
    """
    return n % 2 != 0


def is_teenager(age: int) -> bool:
    """Return whether age is between 13 and 18 inclusive.

    HINT: identify the range of integers that make this function return True.

    if age < 13:
        return False
    else:
        if age > 18:
            return False
        else:
            return True
    """
    return age >= 13 and <= 18


def is_common_prefix(prefix: str, s1: str, s2: str) -> bool:
    """Return whether prefix is a common prefix of both s1 and s2.

    if str.startswith(s1, prefix):
        if str.startswith(s2, prefix):
            return True
        else:
            return False
    else:
        return False
    """
    return str.startswith(s1, prefix) and str.startswith(s2, prefix)


def same_corresponding_values(mapping: dict, key1: str, key2: str) -> bool:
    """Return whether the two given keys have the same corresponding value in mapping.

    Return False if at least one of the keys is not in the mapping.

    if key1 not in mapping:
        return False
    elif key2 not in mapping:
        return False
    elif mapping[key1] == mapping[key2]:
        return True
```

```
    else:
        return False
    """
```

4 Ex 4: The order of if conditions

Consider the following two functions:

```
def pick_animal1(number: int) -> str:
    """Return an animal (based on a number range)."""
    if number > 1:
        return 'Cat'
    elif number > 10:
        return 'Dog'
    else:
        return 'Duck'


def pick_animal2(number: int) -> str:
    """Return an animal (based on a number range)."""
    if number > 10:
        return 'Cat'
    elif number > 1:
        return 'Dog'
    else:
        return 'Duck'
```

We will now use our knowelede of comprehenstions, `range`, and, `all` to write expressions that descirbe the return values these two functoins.

1. Fill in the blanks in each of the following code snippets to satisfy the given description.

   (a) A set of 3 integers where `pick_animal1` only returns `'Duck'`:

   ```
   >>> all([pick_animal1(x) == 'Duck' for x in {-1, -2, -3}])
   True
   ```

   (b) A range of at least 3 integers wehre `pick_animal1` only returns `'Cat'`:

   ```
   >>> all([pick_animal1(x) == 'Duck' for x in {2, 3, 4}])
   True
   ```

   (c) A set of 3 integers where `pick_animal2` only returns `'Duck'`:

   ```
   >>> all([pick_animal2(x) == 'Duck' for x in {-1, -2, -3}])
   True
   ```

   (d) A range of at least 6 integers where `pick_animal2` only returns `'Cat'`:

   ```
   >>> all([pick_animal2(x) == 'Cat' for x in range(11, 20)])
   True
   ```

   (e) The largest range of integers where `pick_animal2` only returns `'Dog'`:

```
>>> all([pick_animal2(x) == 'Dog' for x in range(2, 10)])
True
```

2. Can `pick_animal1` ever return `'Dog'`? Why or why not

   No. If a number is greater than 10, then it is also greater than one. When the first branch of the if statment gets triggered, the interpreter ignores the rest of the branches in that block.