CSC110 Lecture 11: For Loops

Hisbaan Noorani

October 06, 2020

Contents

1 Ex 1: Practice with for loops

1. Consider the following function.

```
1  def sum_of_squares(numbers: List[int]) -> int:
2      """Return the sum of the squares of the given numbers.
3
4      >>> sum_of_squares([4, -2, 1])  # 4 ** 2 + (-2) ** 2 + 1 ** 2
5      21
6      """
7      sum_so_far = 0
8
9      for number in numbers:
10         sum_so_far = sum_so_far + number ** 2
11
12     return sum_so_far
```

   (a) What is the loop variable?

   `number`

   (b) What is the accumulator?

   `sum_so_far`

   (c) Fill in the loop accumulation table for the call to function ~$\text{sum}_{\text{ofsquares}}$([4, -2, 1]).

Iteration	Loop Variable	Loop accumulator
0	-	0
1	4	16
2	-2	20
3	1	21

2. Implement the following function.

```python
def long_greeting(names: List[str]) -> str:
    """Return a greeting message that greets every person in names.

    Each greeting should have the form "Hello <name>! " (note the space at the end).
    The returned string should be the concatenation of all the greetings.

    >>> long_greeting(['David', 'Mario'])  # Note the "extra" space at the end
    'Hello David! Hello Mario! '
    """
    greeting = ''

    for name in names:
        greeting = greeting + 'Hello ' + name + '! '

    return greeting
```

2 Ex 2: Marriage license data revisisted.

In Lecture 9, we saw how to query marriage license data using a nested list (i.e., `List[list]`). In Lecture 10, we saw how to use data classes to store the marriage license data using a list of `MarriageData` (i.e., `List[MarriageData]`):

```python
@dataclass
class MarriageData:
    """..."""
    id: int
    civic_centre: str
    num_licenses: int
    month: datetime.date
```

Each of the following sets of functions takes marriage license data and performs some aggregation of those values. The first function in each set is implemented for you and uses a nested list (as we did at the end of last week). Your task is to implement each of the other two functions in the set in two ways: first with a comprehension, and second with a for loop.

1. Group 1.

```python
def total_licenses_for_centre_v1(data: List[list], civic_centre: str) -> int:
    """Return how many marriage licenses were issued in the given civic centre."""
    return sum([row[2] for row in data if row[1] == civic_centre])
```

```python
def total_licenses_for_centre_v2(data: List[MarriageData], civic_centre: str) -> int:
    """Return how many marriage licenses were issued in the given civic centre."""
    return sum([row.num_licenses for row in data if row.civic_centre == civic_centre])
```

```python
def total_licenses_for_centre_v3(data: List[MarriageData], civic_centre: str) -> int:
    """Return how many marriage licenses were issued in the given civic centre."""
    total = 0

    for row in data:
        if row.civic_centre == civic_centre:
            total = total + row.num_licenses
```

```
    return total
```

2. Group 2.

```
def civic_centre_meets_threshold_v1(data: List[list], civic_centre: str, num: int) -> bool:
    """Return whether civic_centre issued at least num marriage licences every month.

    You only need to worry about the rows that appear in data; don't worry about "missing" months.

    Preconditions:
        - num > 0
        - civic_centre in {'TO', 'NY', 'ET', 'SC'}
        - data satisfies all of the properties described in Worksheet 9, Exercise 2
    """
    licenses_issued = [row[2] for row in data if row[1] == civic_centre]
    return all(num_issued >= num for num_issued in licenses_issued)
```

```
def civic_centre_meets_threshold_v2(data: List[MarriageData], civic_centre: str, num: int) -> bool:
    """Return whether civic_centre issued at least num marriage licences every month.

    You only need to worry about the rows that appear in data; don't worry about "missing" months.

    Preconditions:
        - num > 0
        - civic_centre in {'TO', 'NY', 'ET', 'SC'}
    """
    licenses_issued = [row.num_licenses for row in data if row.civic_centre == civic_centre]
    return all(num_issued >= num for num_issued in licenses_issued)
```

```
def civic_centre_meets_threshold_v3(data: List[MarriageData], civic_centre: str, num: int) -> bool:
    """Return whether civic_centre issued at least num marriage licences every month.

    You only need to worry about the rows that appear in data; don't worry about "missing" months.

    Preconditions:
        - num > 0
        - civic_centre in {'TO', 'NY', 'ET', 'SC'}
    """
    for row in data:
        if row.civic_centre == civic_centre and row.num_licenses < num:
            return False

    return True
```

3 Ex 3: Looping over other data types

For each function, add at least one example to the docstring and complete the function body.

1. One

```python
def count_uppercase(s: str) -> int:
    """Return the number of uppercase letters in s.

    >>> count_uppercase('HELLO')
    5

    >>> cout_uppercase('Hisbaan')
    1
    """
    total_upper = 0

    for character in s:
        if str.isupper(character)
            total_upper = total_upper + 1

    return total
```

2. Two

```python
def all_fluffy(s: str) -> bool:
    """Return whether every character in s is fluffy.

    Fluffy characters are those that appear in the word 'fluffy'.

    >>> all_fluffy('fluffy')
    True

    >>> all_fluffy('fffff')
    True

    >>> all_fluffy('hello)
    False
    """
    for character in s:
        if character not in 'fluffy':
            return False

    return True
```

3. Three

```python
def sum_davids(scores: Dict[str, int]) -> int:
    """Return the sum of all values in scores that correspond to a key that contains 'David'.

    >>> I can't be bothered to type out a dictionary.
    """
    sum = 0

    for score in scores:
        if 'David' in score:
            sum = sum + scores[score]

    return sum
```

4. Four

```
def david_vs_mario(scores: Dict[str, int]) -> str:
    """Return the name of the person with the highest total score in scores.

    David's score is the sum of all values in scores that correspond
    to a key that contains the string 'David'.

    Mario's score is the sum of all values in scores that correspond
    to a key that contains the string 'Mario'.

    >>> I can't be bothered to type out a dictionary.
    """
    mario_score = 0
    david_score = 0

    for mario in scores:
        if 'Mario' in mario:
            mario_score = mario_score + scores[mario]

    for david in scores:
        if 'David' in david:
            david_score = david_score + scores[mario]

    if mario_score >= david_score:
        return 'Mario'
    else:
        return 'David'
```

4 Additional Exercises

Implement the function below in two ways: first using comprehensions, and second using a for loop.

```
def count_anomalies(data: List[MarriageData]) -> int:
    """Return the number of months where there is at least one
    civic centre differing by at least 100 from the average number
    of marriage licenses.
    """
```