CSC110 Lecture 3: Functions

Hisbaan Noorani

September 15, 2020

Contents

1 Ex 1: Practice with built-in functions

1. Suppose we have executed the following assignment statements in the python console:

```
>>> n = -5
>>> numbers_list = [1, 10, n]
>>> numbers_set = {100, n, 200}
```

Write down what each of the following expressions evalusate to. Do this by hand first! (Then check your work in the python console.)

```
>>> abs(n)
5

>>> sorted(numbers_list)
[-5, 1, 10]

>>> sorted(numbers_set) + sorted(numbers_list)
[-5, 100, 200, -5, 1, 10]

>>> type(numbers_set)
<class 'set'>

>>>type(numbers_list == n)
<class 'bool'>

>>> sum(numbers_set) - n
300

>>> max(numbers_list + [5])
10
```

2 Ex 2: Practice with methods

Note: you might want to use Section A.2 Python Build-in Data Types REference as a reference in this exercise.

1. Suppose we have executed the following assignment statemnt in the Python console:

```
>>> wish = 'Happy Birthday'
```

   Write down what each of the following expressions evaluate to. Do this by hand first! (Then check your work in the python console.)

```
>>> str.lower(wish)
'happy birthday'

>>> str.lower(wish[0]) + str.lower(wish[6])
'hb'

>>> str.isalnum(wish)
False

>>> str.count(wish, 'y')
2
```

2. Suppose we have executed the following assignment statments in the Python console:

```
>>> set1 = {1, 2, 3}
>>> set2 = {2, 4, 5, 10}
>>> set3 = {'cat', 'dog', 'rabbit'}
```

   Write down what each of the following expressions evaluate to. Don't worry about the order of elements in a set. Do this by hand first! (Then check your work in the Python console.)

```
>>> set.union(set1, set2)
{1, 2, 3, 4, 5, 10}

>>> set.intersection(set1, set2)
{2}

>>> set.intersection(set1, set3)
{}

>>> set.difference(set3, set2)
{'cat', 'dog', 'rabbit'}

>>> set.difference(set1, set2)
{1, 3}

>>> set.symmetric_difference(set1, set2)
{1, 3, 4, 5, 10}
```

3 Ex 3: Functoins definitons in Python

1. Answer the questions below about the following function definiton.

```
def calculate(x: int, y: int) -> list:
    """Return a list containing the sum and product of the two given numbers.
    """
    return [x + y, x * y]
```

(a) What is the function header?
def calculate(x: int, y: int) -> list:

(b) What is the function name?
calculate

(c) How many parameters does this function have? What are their names and types?
2, x -> int, y -> int

(d) What is the function's return type?
list

(e) What isthe part surrounded by tripple-quotes (“”“) called? What is it's purpose?
Documentation. It describes what the function does.

(f) What is the function body?
return [x + y, x * y]

(g) Compared to the examples we looked at in lectures, what part is missing from this function definiton?
Examples in the documentation.

(h) Write down what you would add to complete this funciton definiton.

```
    >>> calculate(0, 0)
    [0, 0]

    >>> calculate(1, 1)
    [2, 1]
```

2. For each of the function definitons given below, complete the definition by writing a description and one doctest example.

```
def is_same_length(list1: list, list2: list) -> bool:
    """Return whether both lists are the same length.
    >>> is_same_length([1, 2, 3, 4, 5], [1, 2, 3, 4])
    False
    >>> is_same_length([1, 2, 3], [3, 3, 3])
    True
    """
    return len(list1) == len(list2)

def multiply_sums(set1: set, set2: set) -> int:
    """Return an integer that is equal to the product of the sum of the sets.
    >>> multiply_sums({1, 2}, {3, 4})
    21
    """
    return sum(set1) * sum(set2)

```

```
    def exponentiate(nums: list) -> list:
        """Return the square of every element in the list.
        >>> exponentiate([1, 2, 7])
        [1, 4, 59]
        """
        return [x ** x for x in nums]
```

3. Complete each of the following functions according to their docstring description and doctests.

```
    def different_sums(set1: set, set2: set) -> bool:
        """Return whether set1 and set2 have different sums.
        >>> different_sums({1, 2, 3}, {5, -1})
        True
        >>> different_sums({3}, {1, 2})
        False
        """
        return sum(set1) != sum(set2)
```

```
    def squares(n: int) -> dict:
        """Return a dictionary mapping the numbers from 1 to n to their squares.
        Assume that n > 1.
        >>> squares(3)
        {1: 1, 2: 4, 3: 9}
        """
        return {x: x ** 2 for x in range(n)}
```

4 Ex 4: Function scope and vairables

1. Consider the following code snippet:

```
def add_together(x: int, y: int) -> int:
    return x + y


eleven = add_together(5, 6)
twelve = x + 7
```

Running the code snippets produces the following output:

```
Traceback (most recent call last):
    File "path/to/example-name-error-1.py", line 6, in <module>
        twelve = x + 7
NameError: name 'x' is not defined
```

(a) Which line of code is causing the error?
`twelve = x + 7`

(b) What does `NameError: Name 'x' is not defined` mean?
x is not in the scope where it is called. It exists only inside of the function.

(c) Why is `x` not defined?
`x` only exsits within the function `add_together`. It is not accessable from outside of that.

2. Recall the functions square and `calculate_distance`, which some "markers" added:

```
    def square(x: float) -> float:
        """Return x squared.

        >>> square(3.0)
        9.0
        >>> square(2.5)
        6.25
        """
        # MARKER A
        return x ** 2
```

```
    def calculate_distance(p1: tuple, p2: tuple) -> float:
        """Return the distance between points p1 and p2.

        p1 and p2 are tuples of the form (x, y), where the x- and y-coordinates are points.

        >>> calculate_distance((0, 0), (3.0, 4.0))
        5.0
        """
        x1 = p1[0]
        y1 = p1[1]
        x2 = p2[0]
        y2 = p2[1]
        # MARKER B
        return (square(x1 - x2) + square(y1 - y2)) ** 0.5
```

Suppose we run the following in the python conose:

```
    >>> p1 = (0, 0)
    >>> p2 = (5, 3)
    >>> calculate_distance(p1, p2)
```

(a) Draw a value-based memory model diagram to show the current state of the variables when "MARKER B" is reached. Make sure to show both the vriables in the console (`__main__`) and the variables in `calculate_distance`.

`console`:

Variable	Value
p1	(0, 0)
p2	(5, 3)

`calculate_distance`:

Vairable	Value
p1	(0, 0)
p2	(5, 3)
x1	0
y1	0
x2	5
y2	3

`square - first call:`

Variable	Value
x	-5

`square - second call:`

Vairable	Value
x	2