National University of Singapore School of Computing CS1010X: Programming Methodology Semester II, 2024/2025

Recitation 5 Working with Sequences

Python

- 1. Equality Testing
 - == returns True if two objects are equivalent.
 - is returns True if two objects are identical, i.e. they are the same object.
- 2. Membership Testing
 - x in y returns True if x is contained in the sequence type y. There are three basic sequence types in Python: tuples, lists, and range objects. We will discuss lists after the midterm break.
- 3. Range Type
 - The range type represents an immutable sequence of numbers and is commonly used for looping a specific number of times in for loops.
 - range(start, stop[, step]): Creates a sequence starting at start, at intervals of step, up to (but not including stop). If step is zero, ValueError will be raised.
 - range(stop): Creates a sequence starting at 0, at intervals of 1, up to (but not including stop).
 - The advantage of the range type over a regular tuple is that a range object will always take the same (small) amount of memory, no matter the size of the range it represents (as it only stores the start, stop and step values, calculating individual items and subranges as needed), i.e. O(1) space.

Problems

1. Evaluate the following expressions:

```
1 == 1
1 is 1
"foo" == 'foo'
0 == "0"
0 is "0"
False == False
```

```
False == 0
(1, 2) == (1, "2")
(1, 2) is (1, 2)
(1, 2, 3, 4, 5) == (1, 2, 3, 4, 5)
(1, 2, 3, 4, 5) == (1, 2, 3, 5, 4)
((1, 2), (2, 3)) == ((1, 2), (3, 2))
x = (1, 2)
y = (1, 2)
z = x
x is y
x == y
x is z
3 in (1, 2, 3, 4, 5)
(1, 2) in (1, 2, 3, 4, 5)
(2) in (1, 2, 3, 4, 5)
() == ()
(1) == 1
(1, ) == 1
a = ((1,2), (3,4))
b = (x, (3,4))
x in a
x in b
```

2. The function in will test if an object is inside a tuple by checking for equivalence. Write a function contains that will check if an object is inside a tuple by checking for identity. For example,

```
x = (1,2)

a = ((1,2), (3,4))

b = (x, (3,4))

contains(x,a) => False
```

```
contains(x,b) => True
```

Write a function deep_contains that will check if an object is nested arbitrarily deep within a tuple. For example,

```
x = (1,2)
a = ((1,2), ((3,4), x), (5,6))
contains(x,a) => False
deep_contains(x,a) => True
```

3. The accumulate procedure discussed in lecture is also known as fold_right, because it combines the first element of the sequence with the result of combining all the elements to the right. There is also a fold_left, which is similar to fold_right, except that it combines elements working in the opposite direction:

```
def accumulate(fn, initial, seq):
    if seq == ():
        return initial
    else:
        return fn(seq[0], accumulate(fn, initial, seq[1:]))

def fold_left(fn, initial, seq):
    if seq == ():
        return initial
    else:
        return fn(fold_left(fn, initial, seq[:-1]), seq[-1])
```

(a) Suppose we also define the following functions:

```
def pair(a, b):
    return (a, b)

def divide(a, b):
    return a/b

What are the values of

fold_right(divide, 1,(1, 2, 3))
fold_left(divide,1,(1, 2, 3))
```

fold_right(pair,(),(1, 2, 3))
fold_left(pair,(),(1, 2, 3))

- (b) Give a property that fn should satisfy to guarantee that fold_right (or accumulate) and fold_left will produce the same values for any sequence.
- 4. A *queue* is a data structure that stores elements in order. Elements are enqueued onto the tail of the queue. Elements are dequeued from the head of the queue. Thus, the first element enqueued is also the first element dequeued (FIFO, first-infirst-out). The ghead operation is used to get the element at the head of the queue.

```
qhead(enqueue(5, empty_queue()))
# Value: 5

q = enqueue(4, enqueue(5, enqueue(6, empty_queue())))
qhead(q)
# Value: 6

qhead(dequeue(q))
# Value: 5
```

- (a) Decide on an implementation for queue.
- (b) Implement empty_queue

```
def empty_queue():
```

Order of growth in time?

Space?

(c) Implement enqueue; a function that returns a new queue with the element x added to the tail of q.

```
def enqueue(x, q):
```

Order of growth in time?

Space?

(d) Implement dequeue; a function that returns a new queue with the head element removed from q.

```
def dequeue(q):
```

Order of growth in time?

Space?

(e) Implement qhead; a function that returns the value of the head element of q.

def qhead(q):

Order of growth in time?

Space?

- 5. **Homework:** Suppose x is bound to the tuple (1, 2, 3, 4, 5, 6, 7). Using map, filter, accumulate and/or lambdas (as discussed in Lecture), write an expression involving x that returns:
 - (a) (1, 4, 9, 16, 25, 36, 49)
 - (b) (1, 3, 5, 7)
 - (c) ((1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7))
 - (d) ((2, 4), 6)
 - (e) (((2,), 4), 6)
 - (f) The maximum element of x: 7
 - (g) The minimum element of x: 1
 - (h) The maximum squared even element of x: 36
 - (i) The sum of the square of each value in x: 140

You are encouraged to provide multiple solutions for the above questions.