CSC148 winter 2017

code tracing, oddities, review week 12

Danny Heap
heap@cs.toronto.edu
BA4270 (behind elevators)
http://www.cdf.toronto.edu/~csc148h/winter/
416-978-5899

April 5, 2017

Outline

tracing code

aliasing

summary/review

know your code

...inside out, left to right

```
def f(n):
    return n + 1

def g(m):
    return f(m) + 1

f(g(f(1) * 2) + 2)
```

visualize composed functions

recursive

```
def fibonacci(n):
    """explore fibonacci"""
        if n < 2:
            return n
        else:
            return fibonacci(n - 1) + fibonacci(n - 2)
fibonacci(4)
```

visualize fibonacci

memoized fibonacci

```
def fib_memo(n, seen):
    if n not in seen:
        seen[n]  = (n if n < 2
                    else fib_memo(n - 2, seen) + fib_memo(n - 1 , seen)
    return seen[n]
```

visualize fib_memo

know more code

... bottom to top (of hierarchies)

```
class A:
    def g(self, n):
        return n + 1

    def f(self, m):
        return self.g(m)

class B(A):
    def g(self, n):
        return 2 * n

if __name__ == "__main__":
    b = B()
    print(b.f(2))
    a = A()
    print(a.f(2))
```

visualize hierarchy

...and more code

...think locally...

```
x = 7

def f():
    y = x
    print(y)
    if False:
        x = 2

if __name__ == "__main__":
    f()
```

visualize scope !?*

aliases

```
>>> L = [[]] * 3
>>> L
[[], [], []]
>>> L[0].append(1)
>>> L
```

visualize aliases...

persistent values

default values are created when a function is defined...!

```
>>> def f(n, m=[]):
...     m.append(n)
...     return m
...
>>> f(1)
[1]
>>> f(2)
[1, 2]
```

visualize defaults

linked list redux...

```
def __init__(self, value, nxt=None, prev=None):
    """
    Create LinkedListNode self with data value, successor nxt, and
    predecessor prev.

    @param LinkedListNode self: this LinkedListNode
    @param object value: data of this linked list node
    @param LinkedListNode|None nxt: successor to this LinkedListNod
    @param LinkedListNode|None prev: predecessor to this LinkedList
    @rtype: None
    """
    self.value, self.nxt, self.prev = value, nxt, prev
    if self.prev is not None:
        self.prev.nxt = self
    if self.nxt is not None:
        self.nxt.prev = self
```

generators

```
class MyRange:
    """ range implemented using generator pattern.
    """
    def __init__(self, target):
        self.target, self.num = target,0

    def __iter__(self):
        return self

    def __next__(self):
        if self.num < self.target:
            current, self.num = self.num, self.num+1
            return current
        else:
            raise StopIteration()
```

bare bones

- 3 hours
- 7 questions
- comprehensive
- no aid sheet

PLEASE HAND IN

UNIVERSITY OF TORONTO
Faculty of Arts and Science

April 2017 Examinations

CSC 148H1S
Duration — 3 hours
No aids allowed.

PLEASE HAND IN

Student Number: |_|_|_|_|_|_|_|_|_|_|_|_|_|

Last Name: _____

First Name: _____

Do **not** *turn this page until you have received the signal to start.*
(In the meantime, please fill out the identification section above, and read the instructions below.)

This exam consists of 7 questions on 22 pages (including this one). *When you receive the signal to start, please make sure that your copy of the exam is complete.*

Please answer questions in the space provided. If you need additional space, clearly indicate on the question page where to find your answer.

You will earn 20% for any question you leave blank or write "I cannot answer this question" on. You may earn substantial part marks for writing down the outline of a solution and indicating which steps are missing.

You must achieve 40% of the marks on this final exam to pass this course.

There is a Python API at the end of this exam.

1: _____/ 8
2: _____/ 8
3: _____/10
4: _____/ 6
5: _____/ 6
6: _____/10
7: _____/ 8

TOTAL: _____/56

Good Luck!

Short Python function/method descriptions, and classes

```
__builtins__:
  len(x) -> integer
    Return the length of the list, tuple, dict, or string x.
  max(L) -> value
    Return the largest value in L.
  min(L) -> value
    Return the smallest value in L.
  range([start], stop, [step]) -> list of integers
    Return a list containing the integers starting with start and
    ending with stop - 1 with step specifying the amount to increment
    (or decrement). If start is not specified, the list starts at 0.
    If step is not specified, the values are incremented by 1.
  sum(L) -> number
    Returns the sum of the numbers in L.

dict:
  D[k] -> value
    Return the value associated with the key k in D.
  k in d -> boolean
    Return True if k is a key in D and False otherwise.
  D.get(k) -> value
    Return D[k] if k in D, otherwise return None.
  D.keys() -> list of keys
    Return the keys of D.
  D.values() -> list of values
    Return the values associated with the keys of D.
  D.items() -> list of (key, value) pairs
    Return the (key, value) pairs of D, as 2-tuples.

float:
  float(x) -> floating point number
    Convert a string or number to a floating point number, if
    possible.

int:
  int(x) -> integer
    Convert a string or number to an integer, if possible. A floating
    point argument will be truncated towards zero.

list:
  x in L -> boolean
    Return True if x is in L and False otherwise.
  L.append(x)
    Append x to the end of list L.
  L1.extend(L2)
    Append the items in list L2 to the end of list L1.
  L.index(value) -> integer
    Return the lowest index of value in L.
```

```
  L.insert(index, x)
    Insert x at position index.
  L.pop()
    Remove and return the last item from L.
  L.pop(i)
    Remove and return L[i]
  L.remove(value)
    Remove the first occurrence of value from L.
  L.sort()
    Sort the list in ascending order.

Module random:
  randint(a, b)
    Return random integer in range [a, b], including both end points.

str:
  x in s -> boolean
    Return True if x is in s and False otherwise.
  str(x) -> string
    Convert an object into its string representation, if possible.
  S.count(sub[, start[, end]]) -> int
    Return the number of non-overlapping occurrences of substring sub
    in string S[start:end]. Optional arguments start and end are
    interpreted as in slice notation.
  S.find(sub[,i]) -> integer
    Return the lowest index in S (starting at S[i], if i is given)
    where the string sub is found or -1 if sub does not occur in S.
  S.split([sep]) -> list of strings
    Return a list of the words in S, using string sep as the separator
    and any whitespace string if sep is not specified.

set:
  {1, 2, 3, 1, 3} -> {1, 2, 3}
  s.add(...)
    Add an element to a set
  {1, 2, 3}.union({2, 4}) -> {1, 2, 3, 4}
  {1, 2, 3}.intersection({2, 4}) -> {2}
  set()
    Create a new empty set object
  x in s
    True iff x is an element of s

list comprehension:
   [<expression with x> for x in <list or other iterable>]

functional if:
   <expression 1> if <boolean condition> else <expression 2>
   -> <expression 1> if the boolean condition is True,
      otherwise <expression 2>
```

topics

object-oriented programming and design: lecture slides and example code, in-class exercises, weeks 1–2, lab #1, lab #2, and assignment #1

abstract data types, stacks, queues: lecture slides and example code, week 3, lab #3

reading, writing recursion on nested lists: lecture slides and example code, in-class exercise, week 4, lab #4

modularity, functional programming idiom lecture slides and example code, week 5, lab #5

linked lists: lecture slides and example code, in-class exercise, week 6, lab #6

more topics

recursion on general trees: lecture slides and example code week 7, in-class exercises on contains and leaf, lab #7

recursion on binary trees: lecture slides and example code week 8, lab #8, and assignment #2

binary search trees, insertion, deletion, mutation: lecture slides and example code week #9, in-class exercise

efficiency, recursion, recursive structures: lecture slides and example code week #10, lab #9

big-Oh, big-Theta, hash table: lecture slides and example code week #11

hash table, tracing and traps: lecture slides and example code week #12

office hours: Mondays 2:30–5 p.m., April 10, 17, 24

some more prep:

Exam jam:
http://www.artsci.utoronto.ca/current/exam_jam
... puppies, practice, etc.
Extra lab http://www.teach.cs.toronto.edu/~csc148h/winter/Labs/lab10/handout.pdf — no marks, no TAs, no physical lab. Some practice on complexity.

old exam question...

```
def swap_even(t, depth=0):
    """ Swap left and right children of nodes at even depth.
    ... # stuff omitted for space...
    >>> b1 = BinaryTree(1, BinaryTree(2, BinaryTree(3)))
    >>> b4 = BinaryTree(4, BinaryTree(5), b1)
    >>> print(b4)
        1
            2
                3
    4
        5
    >>> swap_even(b4)
    >>> print(b4)
        5
    4
        1
                3
            2
    """
```