CS1210 Computer Science I: Foundations

Exam 1

Thursday, October 1, 2015

This test consists of 3 sections, all with multiple items, worth a total of 70 points. Because they are of varying difficulty, it is probably good strategy not to get stuck too long on any one question. You have 90 minutes: **please write legibly** and **don't write in tiny letters**. Remember, real people with poor eyesight (*i.e.*, me) have to read it.

1. Expressions, Assignments and the Python REPL (46 points)

Evaluate each of the following expressions. If an expression cannot be evaluated (*i.e.*, it yields an error) explain why. Show any output that would be shown by the Python REPL. Moreover, assume the following variables have the specified values:

```
L = [3, 5, 4, 2, 9, -4, 6]
W = ['noble', 'chuck', 'could', 'nosey', 'bears', 'those']
S = 'How much wood could a woodchuck chuck if a woodchuck could chuck wood'
```

(1) `>>> len(W[3])*'3'`

Answer:
'33333' *(n.b., was generous with missing quotes)*

(2) `>>> 'UI'>'ISU'`

Answer:
True

(3) `>>> True and 3=3`

Answer:
SyntaxError: can't assign to operator

(4) `>>> W.index('could')*4`

Answer:
8

(5) `>>> S.split()[W.index('could')+1][-3:-1]`

Answer:
'ul'

(6) `>>> S[12:][:-12].title().count('d')`

Answer:
3

(7) >>> print('this is \not \\a \\\test')

Answer:
this is
ot \a \   est

(8) >>> range(1,11,2)[3:]

Answer:
range(7, 11, 2) *(n.b., give partial credit for [7, 9] or (7, 9))*

(9) >>> 1,245,455

Answer:
(1, 245, 455)

(10) >>> 0-1---2--3--4-5---6//2

Answer:
-4

(11) >>> tuple(list((1)))

Answer:
TypeError: 'int' object is not iterable

(12) >>> False or False and True or not True

Answer:
False

(13) >>> {1, 2, 3} ˆ {3, 4, 1} | {3}

Answer
{2, 3, 4} *(n.b., ˆ is symmetric set difference)*

(14) >>> { x for x in W for y in S.lower().split() if x in y }

Answer:
{'could', 'chuck'}

(15) >>> ' '.join([w[1:len(w)-1] for w in S.split()[-5:11] if len(w)>2])

Answer: 'oodchuc oul' else 'oodchucoul' if join on null also allowed.

(16) >>> tuple({ S.count(x) for x in [w for w in S.lower().split() if ’o’ not in w]})

Answer:
(1, 2, 4)

(17) >>> (W[-1],) + ((W[0],) + tuple(W[2:-1][1:]))

Answer:
(’those’, ’noble’, ’nosey’, ’bears’)

(18) >>> { ’h’:2, ’a’:12, ’e’:-1, ’k’:3 }["hawkeye"[3]] + "hawkeye".index(’w’)

Answer:
5 *(n.b., this is the same as quiz 1)*

(19) >>> [{x, y, z} for x in range(2) for y in range(2) for z in range(2)]

Answer:
[{0}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {0, 1}, {1}] *(n.b., sets cull duplicates)*

(20) >>> ’’.join([W[i][-i-1] for i in range(len(W)-2,-1,-1)])

Answer:
’bouce’

Each of the following expressions set or modify a variable value. If an expression cannot be evaluated (*i.e.*, it yields an error) explain why and show the new value, if any, of the variable on the left hand side of the equation. You should assume that the values of W, S and L are carried over from the previous section.

(1) >>> L[4:] = [-1]*2

Answer:
L is now [3, 5, 4, 2, -1, -1]

(2) >>> W[0:2][-1] = ’abbey’

Answer:
W is still [’noble’, ’chuck’, ’could’, ’nosey’, ’bears’, ’those’]
(n.b., W[0:2] is a copy of part of W)

(3) >>> S[14:19] = ’can’

Answer:
TypeError: str object does not support item assignment
(n.b., strings are immutable)

2. Control Statements, Functions and Scoping (14 points)

Consider the following interative session with the Python REPL. What is output to the screen after each input?

```
>>> i = 5
>>> while i > 1:
        print(i)
        if (i%2):
            i = i+1
            print(i)
        i = i/2
    else:
        print(i)
```

Answer:
5
6
3.0
4.0
2.0
1.0 *(n.b., i%2 is an int, which has a Boolean interpretation)*

```
>>> for e in [ x+1 for x in range(12,0,-3) if x%2 ]:
        print(2*str(e) + str(e*2))
```

Answer:
101020
448 *(n.b., + is concatentation; we were generous with missing cr/lf or extra quotes)*

```
>>> def test(x=0.0, y=1.0, z=1.0):
        return(int(x) + (y**2//z))
>>> test()
```

1.0

```
>>> test(y=2)
```

4.0

```
>>> test(5.5, z=2)
```

5.0

```
>>> test(1, 2, 2)
```

3 *(n.b., ints passed replace default floats)*

```
>>> L = [1, 2, 3]

>>> def foo():
        global L
        L = [6, 5, 4]
        print(L)

>>> def bar(x):
        L[1] = sum(x)
        print(L)

>>> def baz():
        L = [4, 5, 6]
        print(L)

>>> foo()
```

[6, 5, 4] *(n.b., changes global L)*

```
>>> bar(L)
```

[6, 15, 4] *(n.b., changes global L while x is local)*

```
>>> L
```

[6, 15, 4] or match previous answer

```
>>> baz(L)
```

Error: unexpected arg to baz()

```
>>> L
```

[6, 15, 4] or match two answers back

3. Recursion (10 points)

(1) The *separate*$(s1, s2 = '')$ function takes a string, s1, and returns a new string with all uppercase elements of s1 following all lowercase elements of s1, still in the same order. So, for example:

```
>>> separate('IHateCSoneANDPython')
'ateoneythonIHCSANDP'
```

Note how all the lower case characters come first, while the upper case characters come last, yet their relative order remains unchanged. Unfortunately the definition is incomplete. Identify α and β to complete the definition.

```
def separate(s1, s2=''):
    if len(β) == 0:
        return(α)
    elif s1[0].upper() == s1[0]:
        return(separate(β[1:], α+β[0]))
    else:
        return(β[0] + separate(s1[1:], s2))
```

α = **__s2__** β = **__s1__**

(2) Given that the *separate*$(s1, s2 ='')$ function is recursive, identify the base case(s) and the recursive step(s) in the definition above.

Answer: len(s1)==0 constitutes the base case, while the elif and else clauses are the recursive steps.

(3) Can you construct an informal argument (just a sentence or two) to convince the reader that *separate*() will always terminate?

Answer: In both recursive steps, the call to separate() excludes the first character of s1, so each recursive call is on a string that is is one character shorter. Eventually, s1 will be of length 0 and the base case will trigger.