CPSC 1301, Computer Science I Lab Assignment

Learning Python, 5th Edition, Chapter12 Lab

Run the following commands in your Python interpreter. These are from *Learning Python, 5th Edition*, Chapter12. Submit a transcript of your session as your deliverable for this lab. This should be a plain text file named pythonlab-Chapter12_lastname.txt.

1 Chapter 12

```
>>> if 1:
>>>     print('true')
>>>
>>> if not 1:
>>>     print('true')
>>>     else:
>>>     print('false')
>>>
>>> x = 'killer rabbit'
>>> if x == 'roger':
>>>     print("shave and a haircut")
>>>     elif x == 'bugs':
>>>     print("what's up doc?")
>>>     else:
>>>     print('Run away! Run away!')
>>>
>>> choice = 'ham'
>>> print({'spam': 1.25, # A dictionary-based 'switch'
>>>     'ham': 1.99, # Use has_key or get for default
>>>     'eggs': 0.99,
>>>     'bacon': 1.10}[choice])
>>> if choice == 'spam': # The equivalent if statement
>>>     print(1.25)
>>>     elif choice == 'ham':
>>>     print(1.99)
>>>     elif choice == 'eggs':
>>>     print(0.99)
>>>     elif choice == 'bacon':
>>>     print(1.10)
>>>     else:
>>>     print('Bad choice')
>>>
>>> branch = {'spam': 1.25,
>>>     'ham': 1.99,
>>>     'eggs': 0.99}
>>> print(branch.get('spam', 'Bad choice'))
>>> print(branch.get('bacon', 'Bad choice'))
>>> choice = 'bacon'
```

```
>>> if choice in branch:
>>>     print(branch[choice])
>>>     else:
>>>     print('Bad choice')
>>>
>>>     print(branch[choice])
>>>     except KeyError:
>>>     print('Bad choice')
>>>
>>> [] or 3
>>> [] or {}
>>> 2 and 3, 3 and 2 # Return left operand if false
>>> [] and {}
>>> 3 and []
>>> A = 't' if 'spam' else 'f' # For strings, nonempty means true
>>> A
>>> A = 't' if '' else 'f'
>>> A
>>> ['f', 't'][bool('')]
>>> ['f', 't'][bool('spam')]
```

2 Chapter 13

```
>>> while True:
>>>
>>> x = 'spam'
>>> while x: # While x is not empty
>>>     print(x, end=' ') # In 2.X use print x,
>>>     x = x[1:] # Strip first character off x
>>>
>>> a=0; b=10
>>> while a < b: # One way to code counter loops
>>>     print(a, end=' ')
>>>     a += 1 # Or, a = a + 1
>>>
>>> while True:
>>>     name = input('Enter name:') # Use raw_input() in 2.X
>>>     if name == 'stop': break
>>>     age = input('Enter age: ')
>>>     print('Hello', name, '=>', int(age) ** 2)
>>>
>>> for x in ["spam", "eggs", "ham"]:
>>>     print(x, end=' ')
>>>
>>> sum = 0
>>> for x in [1, 2, 3, 4]:
>>>     sum = sum + x
>>>
>>> sum
>>> prod = 1
>>> for item in [1, 2, 3, 4]: prod *= item
>>>
```

```
>>> p
>>> S = "lumberjack"
>>> T = ("and", "I'm", "okay")
>>> for x in S: print(x, end=' ') # Iterate over a string
>>>
>>> for x in T: print(x, end=' ') # Iterate over a tuple
>>>
>>> T = [(1, 2), (3, 4), (5, 6)]
>>> for (a, b) in T: # Tuple assignment at work
>>>     print(a, b)
>>>
>>> D = {'a': 1, 'b': 2, 'c': 3}
>>> for key in D:
>>>     print(key, '=>', D[key]) # Use dict keys iterator and index
>>>
>>> list(D.items())
>>> for (key, value) in D.items():
>>>     print(key, '=>', value) # Iterate over both keys and values
>>>
>>> for both in T:
>>>     a, b = both # Manual assignment equivalent
>>>     print(a, b) # 2.X: prints with enclosing tuple "()"
>>>
>>> ((a, b), c) = ((1, 2), 3) # Nested sequences work too
>>> a, b, c
>>> for ((a, b), c) in [((1, 2), 3), ((4, 5), 6)]: print(a, b, c)
>>>
>>> for ((a, b), c) in [([1, 2], 3), ['XY', 6]]: print(a, b, c)
>>>
>>> a, b, c = (1, 2, 3) # Tuple assignment
>>> a, b, c
>>> for (a, b, c) in [(1, 2, 3), (4, 5, 6)]: # Used in for loop
>>>     print(a, b, c)
>>>
>>> a, *b, c = (1, 2, 3, 4) # Extended seq assignment
>>> a, b, c
>>> for (a, *b, c) in [(1, 2, 3, 4), (5, 6, 7, 8)]:
>>>     print(a, b, c)
>>>
>>> for all in [(1, 2, 3, 4), (5, 6, 7, 8)]: # Manual slicing in 2.X
>>>     a, b, c = all[0], all[1:3], all[3]
>>>     print(a, b, c)
>>>
>>> items = ["aaa", 111, (4, 5), 2.01] # A set of objects
>>> tests = [(4, 5), 3.14] # Keys to search for
>>>
>>> for key in tests: # For all keys
>>>     for item in items: # For all items
>>>     if item == key: # Check for match
>>>     print(key, "was found")
>>>     break
>>>     else:
>>>     print(key, "not found!")
>>>
```

```
>>> for key in tests: # For all keys
>>>     if key in items: # Let Python check for a match
>>>     print(key, "was found")
>>>     else:
>>>     print(key, "not found!")
>>>
>>> seq1 = "spam"
>>> seq2 = "scam"
>>>
>>> res = [] # Start empty
>>> for x in seq1: # Scan first sequence
>>>     if x in seq2: # Common item?
>>>     res.append(x) # Add to result end
>>>
>>> res
>>> [x for x in seq1 if x in seq2] # Let Python collect results
>>> list(range(5)), list(range(2, 5)), list(range(0, 10, 2))
>>> list(range(-5, 5))
>>> list(range(5, -5, -1))
>>> for i in range(3):
>>>     print(i, 'Pythons')
>>>
>>> X = 'spam'
>>> for item in X: print(item, end=' ') # Simple iteration
>>>
>>> i = 0
>>> while i < len(X): # while loop iteration
>>>     print(X[i], end=' ')
>>>     i += 1
>>>
>>> X
>>> len(X) # Length of string
>>> list(range(len(X))) # All legal offsets into X
>>>
>>> for i in range(len(X)): print(X[i], end=' ') # Manual range/len iteration
>>>
>>> for item in X: print(item, end=' ') # Use simple iteration if you can
>>> S = 'spam'
>>> for i in range(len(S)): # For repeat counts 0..3
>>>     S = S[1:] + S[:1] # Move front item to end
>>>     print(S, end=' ')
>>>
>>> S
>>> for i in range(len(S)): # For positions 0..3
>>>     X = S[i:] + S[:i] # Rear part + front part
>>>     print(X, end=' ')
>>>
>>> L = [1, 2, 3]
>>> for i in range(len(L)):
>>>     X = L[i:] + L[:i] # Works on any sequence type
>>>     print(X, end=' ')
>>>
>>> S = 'abcdefghijk'
>>> list(range(0, len(S), 2))
```

```
>>> for i in range(0, len(S), 2): print(S[i], end=' ')
>>>
>>> S = 'abcdefghijk'
>>> for c in S[::2]: print(c, end=' ')
>>>
>>> L = [1, 2, 3, 4, 5]
>>> for x in L:
>>>     x += 1 # Changes x, not L
>>>
>>> L
>>> x
>>> L = [1, 2, 3, 4, 5]
>>> for i in range(len(L)): # Add one to each item in L
>>>     L[i] += 1 # Or L[i] = L[i] + 1
>>>
>>> L
>>> i = 0
>>> while i < len(L):
>>>     L[i] += 1
>>>     i += 1
>>>
>>> L
>>> L1 = [1,2,3,4]
>>> L2 = [5,6,7,8]
>>> zip(L1, L2)
>>> list(zip(L1, L2)) # list() required in 3.X, not 2.X
>>> zip(L1, L2)
>>> list(zip(L1, L2)) # list() required in 3.X, not 2.X
>>> T1, T2, T3 = (1,2,3), (4,5,6), (7,8,9)
>>> T3
>>> list(zip(T1, T2, T3)) # Three tuples for three arguments
>>> S1 = 'abc'
>>> S2 = 'xyz123'
>>>
>>> list(zip(S1, S2)) # Truncates at len(shortest)
>>> D1 = {'spam':1, 'eggs':3, 'toast':5}
>>> D1
>>> D1 = {}
>>> D1['spam'] = 1
>>> D1['eggs'] = 3
>>> D1['toast'] = 5
>>> keys = ['spam', 'eggs', 'toast']
>>> vals = [1, 3, 5]
>>> list(zip(keys, vals))
>>> D2 = {}
>>> for (k, v) in zip(keys, vals): D2[k] = v
>>>
>>> D2
>>> keys = ['spam', 'eggs', 'toast']
>>> vals = [1, 3, 5]
>>> D3 = dict(zip(keys, vals))
>>> D3
>>> {k: v for (k, v) in zip(keys, vals)}
>>> S = 'spam'
```

```
>>> offset = 0
>>> for item in S:
>>>     print(item, 'appears at offset', offset)
>>>     offset += 1
>>>
>>> S = 'spam'
>>> for (offset, item) in enumerate(S):
>>>     print(item, 'appears at offset', offset)
>>>
>>> E = enumerate(S)
>>> E
>>> next(E)
>>> next(E)
>>> next(E)
>>> [c * i for (i, c) in enumerate(S)]
>>> for (i, l) in enumerate(open('test.txt')):
>>>     print('%s) %s' % (i, l.rstrip()))
>>>
```