COEN 166 Artificial Intelligence Lab

Assignment #1 Sample Report

Kevin Hom 00001268661

**Part 1**

**Exercise 1: Numbers**

>>> a=123+222

>>> print(a)

345

>>> b=1.5\*4

>>> print(b)

6.0

>>> c=2\*\*10 #2 to the power 10

>>> print(c)

1024

>>> import math

>>> print(math.pi)

3.141592653589793

>>> print(math.sqrt(36))

6.0

>>> import random

>>> a = random.random()

>>> print('a=', a)

a= 0.2243187623188152

>>> b = random.choice([1,2,3,4])

>>> print('b=', b)

b= 2

**Exercise 2: Strings**

>>> S=’Spam’ # make a 4-character string, and assign it to a name

>>> len(S) # Length

4

>>> S[0] # the 1st item in S, indexing by zero-based position

‘S’

>>> S[1] # the 2nd item from the left

‘P’

>>> S[-1] # the last item from the end in S

‘M’

>>> S[-2] # the second-to-last item from the end

‘A’

>>> S[len(S)-1]

‘M’

>>> S[1:3] # Slice of S from offsets 1 through 2 (not 3)

‘PA’

>>> S = ‘z’ + S[1:]

>>> S

‘zPAM’

**Exercise 3: Lists**

>>> L=[123, ‘spam’, 1.23] # A list of three different-type objects

>>> len(L) # number of items in the list

3

>>> L[0]

123

>>> L[:-1] # Slicing a list returns a new list

[123, ‘spam’]

>>> L+[4,5,6] # contact/repeat make new lists too

[123, ‘spam’, 1.23, 4, 5, 6]

>>> L\*2 # repeat

[123, ‘spam’, 1.23, 123, ‘spam’, 1.23]

>>> L # we are not changing the original list

[123, ‘spam’, 1.23]

>>> M = [‘bb’, ‘aa’, ‘cc’]

>>> M.sort()

>>> M

[‘aa’, ‘bb’, ‘cc’]

>>> M.reverse()

>>> M   
[‘cc’, ‘bb’, ‘aa’]

>>> M = [[1,2,3], [4,5,6], [7,8,9]] # a list that contains three other lists, 3x3 matrix

>>> M[1] # get row 2

[4,5,6]

>>> M[1][2] # get row 2, then get item 3 within the row

6

>>> diag = [M[i][i] for i in [0, 1, 2]]

>>> diag

[1,5,9]

>>> doubles = [c \* 2 for c in 'spam']

>>> doubles

[‘ss’, ‘pp’, ‘aa’, ‘mm’]

>>> list(range(4)) *# 0..3 (list() required in 3.X*)

[0, 1, 2, 3]

>>> list(range(−6, 7, 2)) *# −6 to +6 by 2 (need list() in 3.X*)

[-6, -4, -2, 0, 2, 4, 6]

>>> [[x \*\* 2, x \*\* 3] for x in range(4)] # Multiple values, “if” filters

[[0,0], [1,1], [4,8], [9,27]]

>>> [[x, x/2, x \* 2] for x in range(−6, 7, 2) if x > 0]

[[2, 1, 4], [4, 2, 8], [6, 3, 12]]

**Exercise 4: Dictionaries**

>>> D = {'food': 'Spam', 'quantity': 4, 'color': 'pink'}

>>> D['food'] *# Fetch value of key 'food'*

‘Spam’

>>> D['quantity'] += 1 *# Add 1 to 'quantity' value*

>>> D

{'food': 'Spam', 'color': 'pink', 'quantity': 5}

>>> D = {}

>>> D['name'] = 'Bob' *# Create keys by assignment*

>>> D['job'] = 'dev'

>>> D['age'] = 40

>>> D

{'age': 40, 'job': 'dev', 'name': 'Bob'}

>>> print(D['name'])

Bob

>>> bob1 = dict(name='Bob', job='dev', age=40) *# Keywords*

>>> bob1

{'age': 40, 'job': 'dev', 'name': 'Bob'}

>>> bob2 = dict(zip(['name', 'job', 'age'], ['Bob', 'dev', 40])) *# Zipping*

>>> bob2

{'age': 40, 'job': 'dev', 'name': 'Bob'}

**Exercise 5: Tuples**

>>> T = (1, 2, 3, 4) *# A 4-item tuple*

>>> len(T) *# Length*

**4**

>> T + (5, 6) *# Concatenation*

(1, 2, 3, 4, 5, 6)

>>> T[0] *# Indexing, slicing, and more*

**1**

>>> T.index(4) *# Tuple methods: 4 appears at offset 3*

**3**

>>> T.count(4) *# 4 appears once*

**1**

>>> T[0] = 2 *# Tuples are immutable*

*...error text omitted...*

TypeError: 'tuple' object does not support item assignment

>>> T = (2,) + T[1:] *# Make a new tuple for a new value*

>>> T

**(2, 2, 3, 4)**

>>> T = 'spam', 3.0, [11, 22, 33]

>>> T[1]

**3.0**

>>> T[2][1]

**22**

**Exercise 6:**

Output:

block2

block1

block0

1.99

**Exercise 7:**

>>> 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

**spampamamm**

>>> a=0; b=10

>>> while a<b: # one way to code counter loops

print(a,end=’ ‘)

a+=1 # or, a = a + 1

**0123456789**

>>> x=10

>>> while x:

x=x-1 # or, x -=1

if x%2 !=0: continue # odd? – skip print

print(x, end=' ')

**86420**

>>> for x in ["spam", "eggs", "ham"]:

print(x, end=' ')

**spameggsham**

>>> sum = 0

>>> for x in [1, 2, 3, 4]:

sum = sum + x

>>> sum

**10**

>>> prod = 1

>>> for item in [1,2,3,4]: prod\*= item

>>> prod

**24**

**Exercise 8:**

8

OkOkOkOk

**Exercise 9:**

**Test1\_module.py**

30

**Test2\_module.py**

15

200

**Test3\_module.py**

30

200

**Exercise 10:**

**Minimax.py**

This program returns the min and max of an array with an arbitrary size

1

6

**Minimax2.py**

This program returns the min and max of an array with an arbitrary size, but has an if statement to check whether the program is being ran as top-level program file.

1

6

**3.**

When running “import minimax” in the python shell, it executes the program. But, when “import minimax2” is executed, it doesn’t print out the min and max of the array since it not ran as top-level program.

**Exercise 11:**

Class1.py

This program prints out:

King Arthur

-5

QQ

-3

QQ  
 -3

spam

The program first defines a class FirstClass with two member functions setdata and display. The object x is part of the FirstClass and it first sets it data to “King Arthur” and -5, which is then printed using the display member function. Then it resets x’s data by overriding the member variables. In the last three lines, it tries to set x.anothername to spam, but doesn’t change anything to class.

Class2.py

10

20

-15

This program declares object z of the Secondclass. It sets its data1 and data2 to 10 and 20 through the inheritance of the first class. This it prints its data and using the member adder function to add both data1 and data2 of the given object and prints the result.

Class3.py

['dev', 'mgr']

('Sue', ['dev', 'cto'])

This program defines a class Person with an initial constructor taking in three parameters. There is a member function info which just returns the name and job of the individual. It sets two objects rec1 and rec2 as constructors and just prints out the jobs of rec1 and the name and job of rec2.

Class\_as\_module.py

['dev', 'mgr']

('Sue', ['dev', 'cto'])

30

('Jane', ['dev', 'mgr'])

35

('Mike', ['dev', 'mgr'])

This program inherits from program class3.py by inheriting the class Person. The first two lines are the print functions called in class3.py which are executed first. Then it sets parameters and data for rec3 and rec4 which is then printed out.