COEN 166 Artificial Intelligence

Lab Assignment #1 Report

Eva Stenberg **00001576722**

Part I

Exercise 1: Numbers

```
a=123+222
print(a)
345

b=1.5*4
print(b)
6.0

c=2**10 # 2 to the power of 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= 0.29761641053575094

b=random.choice([1,2,3,4])
print('b=',b)
b= 4
```

Exercise 2: Strings

```
>>> S='Spam'
>>> len(S)
4
>>> S[0]
'S'
>>> S[1]
'p'
>>> S[-1]
'm'
>>> S[-2]
'a'
>>> S[len(S)-1]
'm'
>>> S[1:3]
'pa'
>>> S='z'+S[1:]
>>> S
'zpam'
```

Exercise 3: List

```
>>> L=[123,'spam',1.23]
>>> len(L)
3
>>> L[0]
123
>>> L[:-1]
[123, 'spam']
>>> L+[4,5,6]
[123, 'spam', 1.23, 4, 5, 6]
>>> L.insert(0,'apple')
['apple', 123, 'spam', 1.23]
>>> L=[123,'spam',1.23]
>>> L.insert(2,'apple')
[123, 'spam', 'apple', 1.23]
>>> L=[1,10,2,5,8]
>>> L.sort()
>>> L
[1, 2, 5, 8, 10]
>>> L.reverse()
>>> L
[10, 8, 5, 2, 1]
```

```
>>> list(range(4))
[0, 1, 2, 3]
>>> list(range(-6,7,2))
[-6, -4, -2, 0, 2, 4, 6]
>>> [[x**2,x**3] for x in range(4)]
[[0, 0], [1, 1], [4, 8], [9, 27]]
>>> [[x, x/2,x*2]for x in range(-6,7,2) if x>0]
[[2, 1.0, 4], [4, 2.0, 8], [6, 3.0, 12]]
```

Exercise 4: Tuple

```
>>> pair=(3,5)
>>> pair[0]
3
>>> x,y=pair
>>> x
3
>>> y
5
```

Exercise 5: if Tests and Syntax Rules

```
x =1
if x:
   y= 2
if y>0:
    print('block2')
    print('block1')
    print('block0')
choice = 'ham'
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')
```

Output:
```
block2
block1
block0
1.99
```

Because x is initialized as equaling 1, y is then set as 2. Because 2 is greater than 0, the function prints block 2, block 1, and block 0. It then initialized choice as “ham” and runs through if else statements and prints 1.99.

Exercise 6: while and for Loops
```
x='spam'
while x:
      print(x,end='')
      x=x[1:]
```
spampamamm
This loop prints the word spam and removes the first letter of the word each time it loops until there are no more letters.

```
a=0; b=10
while a<b:
   print(a,end='')
   a+=1
```
0123456789
This while loop starts at 0 and prints each sequential letter until it reaches 10 because a<b and not a<=b.

```
x=10
while x:
   x=x-1
   if x%2 !=0: continue
   print(x,end='')
```
86420
This while loop starts at 10, goes until 0, and prints every letter that is divisible by 2.

```
thislist = ["spam", "eggs", "ham"]
for x in thislist:
   print(x,end='')
```
spameggsham

This for loop prints each word in the list once.

```
sum = 0
for x in [1,2,3,4]:
    sum = sum+x
print(sum)
```

10

This for loop adds up all the numbers in the list and prints out the final sum.

```
prod = 1
for item in [1,2,3,4]: prod*=item
print(prod)
```

24

This for loop starts at 1 and multiplies the previous element by the next elements until it has traversed the entire list and then stops.

Exercise 7: Functions

```
def times(x,y): # create a function
return x*y # Body executed when called
a = times(2,4)
b = times('Ok', 4) # Functions are “typeless”
print(a,'\n', b)
```

Answer

```
8
 OkOkOkOk
```

Comments

This function is a recursive function that multiplies a and b when called. In the first case, it calls 2 and 4 which produces 8 when they are multiplied. In the second case, because there is a integer y, and a string x, when they are multiplied together, it prints the string y times.

Exercise 8: Modules

module.py:

```
a = 10
b = 20
def adder(x, y): # module attribute
    z = x + y
    return z
def multiplier(x, y):
    z = x*y
    return z
```

Answer

```
30
200
```

test1_module.py:

```
import module # Get module as a whole
result = module.adder(module.a, module.b) # Go through the module name “module” to fetch
# its attributes “adder”, “a”, and “b”
print(result)
```

Answer

```
30
```

test2_module:

```
c = 5
d = 10
from module import adder # Copy out an attribute
result = adder(c, d) # No need go through the module name “module” to fetch
# its attribute “adder”
print(result)
from module import a, b, multiplier # Copy out multiple attributes
result = multiplier(a, b)
print(result)
```

Answer

```
15
200
```

test3_module:

```
from module import* # Copy out all attributes
result1 = adder(a, b)
result2 = multiplier(a, b)
print(result1, '\n', result2)
```

Answer

```
30
200
```

Comments

For **module.py,** a and b were defined integers and the function called them in addition and multiplication resulting in the answers of 30 and 200. In **test1_module.py,** it calls the previous module function and runs the same addition resulting in 30. In **test2_module.py,** it introduces a new c and d, adds them together, and calls the multiplication function from the first module again. In **test3_module,** it calls the initial function again and prints out the former results of addition and multiplication which are 30 and 200.

Exercise 9: built-in attribute of modules

minmax.py:

Answer

```
1
6
```

Comments

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

minmax2.py:

Answer

```
1
6
```

Comments

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

From python shell, execute the following two commands, show the results, and explain your findings.

Answer

```
>>> import minmax
1
6
>>> import minmax2
>>>
```

Comments

When running “import minimax” in the python shell, it executes the program and shows the previously seen 1,6. But, when “import minimax2” is executed, it doesn’t print out the min and max of the array because it is not run as a top-level program.

Exercise 10: Object-Oriented Programming and Classes

class1.py

Answer

```
John
 20

Mary
 30

Mary
 30

Engineering
```

Comments

The program first defines a class FirstClass with two member functions basicinfo and display. It then makes an instance of the class and calls John as Self, which is then printed using the display member function. It sets John’s age as 20 under the variable age. Then it resets the data by changing the name to Mary and giving her an age of 30. It introduces another instance of school and prints that Mary goes to the school of “Engineering”.

class2.py

Answer

```
Cathy
 20

Cathy is a CSE student from Santa Clara University
```

Comments

This program declares a second class. It defines basic info of name and age and then creates more info of the person’s university and major. It then makes an instance and prints its data using the member adder function to add all of the data of the given object and prints the result.

class3.py

Answer

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

Comments

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

class_as_module.py

Answer

```
['dev', 'mgr']
('Sue', ['dev', 'cto'])
30
('Jane', ['dev', 'mgr'])
35
('Mike', ['dev', 'mgr'])
```

Comments

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