Write statement to print second to last element from list L, assuming it has at least 2 items. When L is defined, it would print just letter l.

L = [‘h’, ‘e’, ‘l’, ‘l’, ‘o’]

**Print L[-2]**

After following Python code is executed, what is type of n[:3]?

N = “12345”

**String**

After following Python code is executed, what is type of L[2:3]?

L = [‘h’, ‘e’, ‘l’, ‘l’, ‘o’]

**List**

What will this code print?

Myvar = “hello”

Print myvar[0]

**H**

What will this code print?

Myvar = “hello”

Print “myvar”[0]

**M**

After this code is executed, what is type of variable x?

X = int(12.0)/float(5)

**Float**

What will print?

X = 3

Print x == x+1

**False**

After this is executed, what will be type of variable s?

S = “<published>2009-01-23T20:04:53Z</published>”

T = s.split()

**String & list**

After this is executed, what will be type of variable t?

S = “<published>2009-01-23T20:04:53Z</published>”

T = s.find(“2009”)

**Integer**

What will this print?

S = “<published>2009-01-23T20:04:53Z</published>”

Print s.split(‘-‘)[1]

**01**

What will this print?

S = “<published>2009-01-23T20:04:53Z</published>”

Print len(s.split(‘T’)[0].split(‘2’))

**3**

Consider 2 files, each has lyrics for 1 song. First song has 3 verses, 1 chorus repeated after each verse. Second song has 6 verses, all different from each other. Both files have same # of characters in them. Which file has more info, chorus or no chorus?

**File with no chorus has more info b/c more none repeated info.**

Consider 2 coins, one “fair”, lands on heads/tails 50/50. Other is “biased” which lands heads 75% of time. Which gives more info, fair or biased?

**Fair gives more info- higher info entropy b/c 50% chance instead of 75%/25%.**

You’re connected to directory, ~/Documents/Courses/106/F14/Exams. Write unix command that displays contents of ~/Documents/Courses/106

**Cd ~/Documents/Courses/106 *or* ls ~/Documents/Courses/106/**

For following: L = [“First”, “Second”, “Third”]

What would print?

For x in L:

Y=L[0]

Print y

**First**

**First**

**First**

For x in L:

Y=L[0]

Print y

**First**

For x in L:

Y = x in L

Print y

**True**

**True**

**True**

What will this print?

X= 10.5

If x < 10:

Print “one”

Elif x < 20:

If x > 15:

Print “two”

Else:

Print “three”

Elif x > 0

Print “four”

**Three**

What will this print?

X= 10.5

If x < 10:

Print “one”

Elif x < 20:

If x > 15:

Print “two”

Else:

Print “three”

If x > 0

Print “four”

**Three**

**Four**

What would this print?

L = []

L.append(‘a’)

L.append(‘b’)

L.append(‘c’)

L[1]=0

Print L

**[‘a’, 0, ‘c’]**

What would this print?

D={ }

D[1] = ‘a’

D[2] = ‘b’

D[‘c’]=3

D[‘c’] = d[‘c’]+1

Print d[‘c’]

**4**

What would this print?

D = { }

D[1] = ‘a’

D[2] = ‘b’

D[‘c’] = 3

Print ‘a’ in d.values()

**True**

Write 1 line of code that accomplishes what last three lines of code do

T = (20, 30, 40)

X = t[0]

Y = t[1]

Z = t[2]

**X, y, z = t**

What value prints?

Def g(x, y):

Z = y+x

Return y

Y=10

Z=g(5,y)

Print z

**10**

What will this print?

X = -1

Y = -2

Z = -3

Def h(x, y=2, z=3):

Print x, y, z

H(1)

**1, 2, 3**

Assume following has been executed

L = [{‘a’:1, ‘b’:2, ‘d’:11},

{‘a’:4, ‘b’:5, ‘e’:11},

{‘a’:7, ‘b’:8, ‘f’:11}]

Write code to print each value associated with key ‘b’ in each of dictionaries in L, should print

2

5

8

**for d in L:**

**print d[‘b’]**

Write code that generates a single dictionary with 1 key for each of keys in any of dictionaries in L, and value equal to count of how many dictionaries key appears in. Dictionary it generates should be:

{‘a’:3, ‘b’:3, ‘c’:3, ‘d’:1, ‘e’:1, ‘f’:1}

**dx = {}**

**for d in L:**

**for x in d.keys():**

**if k in dx:**

**dx[k] = dx[k] + 1**

**else:**

**dx[k] = 1**

or

**def new\_dict(lst):**

**dx={}**

**for d in lst:**

**for k in d.keys():**

**if k in dx:**

**dk[k] = dk[k]+1**

**else:**

**dx[k]=1**

**return dx**

**new\_var=new\_dict(L)**

Write code that repeatedly asks user to input numbers. Keep going until sum is 21 or more. Print out total.

**Sum = 0**

**While sum < 21:**

**X = int(raw\_input(“Please enter a number”))**

**Sum = sum + x**

**Print sum**

Define function called deduplicate that takes a list as input and returns a list that has all duplicates removed, keeping only first instance of each item.

Ex: if [1, 2, 3, 2, 4, 2, 3, 4, 5], would return [1, 2, 3, 4, 5]

**Def deduplicate(lst):**

**Acc\_list = []**

**For x in lst:**

**If x not in acc\_lst:**

**Acc\_lst.append(x)**

**Return acc\_lst**