Pre-Lab 05 & 06 Assignments

Table of contents

Anish Goyal 01/13/2023 Stiffler Period 4

Task 0.5.1: Use python to find the number of minutes in a week.

```
print(7*24*60)
```

```
10080
```

Task 0.5.2: Use python to find the remainder of 2304811 divided by 47 without using the modulo operator %. (Hint: Use //.)

```
print(2304811/47-int(2304811/47)*47)
```

```
-2255747.4680851065
```

Task 0.5.3: Enter a Boolean expression to test whether the sum of 673 and 909 is divisible by 3.

```
print((673+909)%3 == 0)
```

```
False
```

Task 0.5.4: Assign the value -9 to x and 1/2 to y. Predict the value of the following expression, then enter it to check your prediction: 2(y+1/2) if x+10<0 else 2(y-1/2)

I believe the value of this expression will be 1.0. This is because $x + 10 \nless 0$ and $2^{\frac{1}{2}-\frac{1}{2}} = 2^0$, which is 1.0. The expression will be evaluated as follows:

```
x = -9
y = 1/2
print(2**(y+1/2) if x+10<0 else 2**(y-1/2))
```

```
1.0
```

Task 0.5.5: Write a comprehension over {1, 2, 3, 4, 5} whose value is the set consisting of the squares of the first five positive integers.

```
print({x**2 for x in {1, 2, 3, 4, 5}})
```

```
{1, 4, 9, 16, 25}
```

Task 0.5.6: Write a comprehension over {0, 1, 2, 3, 4} whose value is the set consisting of the first five powers of two, starting with 2^0.

```
print({2**x for x in {0, 1, 2, 3, 4}})
```

```
{1, 2, 4, 8, 16}
```

Task 0.5.7: The value of the previous comprehension, {x*y for x in {1,2,3} for y in {2,3,4}}, is a seven-element set. Replace {1,2,3} and {2,3,4} with two other three-element sets so that the value becomes a nine-element set.

```
print({x*y for x in {1,2,4} for y in {2,3,5}})
```

```
{2, 3, 4, 5, 6, 8, 10, 12, 20}
```

Task 0.5.8: Replace {1,2,3} and {2,3,4} in the previous comprehension with two dis-joint (i.e. non-overlapping) three-element sets so that the value becomes a five-element set.

```
print({x*y for x in {0,2,4} for y in {8,16,32}})
```

```
{0, 32, 64, 128, 16}
```

Task 0.5.9: Assume that S and T are assigned sets. Without using the intersection operator &, write a comprehension over S whose value is the intersection of S and T. Hint: Use a membership test in a filter at the end of the comprehension. Try out your comprehension with S = {1,2,3,4} and T = {3,4,5,6}.

```
S = {1,2,3,4}
T = {3,4,5,6}
print({x for x in S if x in T})
```

Task 0.5.10: Write an expression whose value is the average of the elements of the list [20, 10, 15, 75].

```
print(sum([20, 10, 15, 75])/len([20, 10, 15, 75]))
```

```
30.0
```

Task 0.5.11: Write a double list comprehension over the lists ['A','B','C'] and [1,2,3] whose value is the list of all possible two-element lists [letter, number]. That is, the value is [['A', 1], ['A', 2], ['A', 3], ['B', 1], ['B', 2],['B', 3], ['C', 1], ['C', 2], ['C', 3]]

```
print([[x, y] for x in ['A','B','C'] for y in [1,2,3]])
```

```
[['A', 1], ['A', 2], ['A', 3], ['B', 1], ['B', 2], ['B', 3], ['C', 1], ['C', 2], ['C', 3]]
```

Task 0.5.12: Suppose LofL has been assigned a list whose elements are themselves lists of numbers. Write an expression that evaluates to the sum of all the numbers in all the lists. The expression has the form sum([sum(... and includes one comprehension. Test your expression after assigning [[.25, .75, .1], [-1, 0], [4, 4, 4, 4]] to LofL. Note that your expression should work for a list of

any length.

```
LofL = [[.25, .75, .1], [-1, 0], [4, 4, 4, 4]]
print(sum([sum(x) for x in LofL]))
```

```
16.1
```

Task 0.5.13: Find out what happens if the length of the left-hand side list does not match the length of the right-hand side list.

```
[x, y, z] = [1, 2]
```

```
ValueError: not enough values to unpack (expected 3, got 2)
```

Task 0.5.14: Suppose S is a set of integers, e.g. {4, 2, 1, 2, 5, 0}. Write a triple comprehension whose value is a list of all three-element tuples (i, j, k) such that i, j, k are elements of S whose sum is zero.

```
S = {4, 2, 1, 2, 5, 0}
print([(i, j, k) for i in S for j in S for k in S if i+j+k == 0])
```

```
[(0, 0, 0)]
```

Task 0.5.16: Further modify the expression so that its value is not the list of all such tuples but is the first such tuple.

```
S = {4, 2, 1, 2, 5, 0}
print([(i, j, k) for i in S for j in S for k in S if i+j+k == 0][0])
```

```
(0, 0, 0)
```

Task 0.5.17: Find an example of a list L such that len(L) and len(list(set(L))) are different.

```
L = [1, 2, 1]
print(len(L))
print(len(list(set(L))))
```

```
3
2
```

Task 0.5.18: Write a comprehension over a range of the form range(n) such that the value of the comprehension is the set of odd numbers from 1 to 99.

```
print({x for x in range(100) if x%2 == 1})
```

```
{1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47,
```

Task 0.5.19: Assign to L the list consisting of the first five letters [‘A’,‘B’,‘C’,‘D’,‘E’]. Next, use L in an expression whose value is [(0, ’A’), (1, ’B’), (2, ’C’), (3, ’D’), (4, ’E’)]. Your expression should use a range and a zip, but should not use a comprehension.

```
L = ['A','B','C','D','E']
print(list(zip(range(5), L)))
```

```
[(0, 'A'), (1, 'B'), (2, 'C'), (3, 'D'), (4, 'E')]
```

Task 0.5.20: Starting from the lists [10, 25, 40] and [1, 15, 20], write a comprehension whose value is the three-element list in which the first element is the sum of 10 and 1, the second is the sum of 25 and 15, and the third is the sum of 40 and 20. Your expression should use zip but not list.

```
print([x+y] for (x, y) in zip([10, 25, 40], [1, 15, 20]))
```

```
<generator object <genexpr> at 0x000001BFC6F86960>
```

Task 0.5.21: Suppose dlist is a list of dictionaries and k is a key that appears in all the dictionaries in dlist. Write a comprehension that evaluates to the list whose ith element is the value corresponding to key k in the ith dictionary in dlist. Test your comprehension with some data. Here are some example data. dlist = [{'James':'Sean', 'director':'Terence'}, {'James':'Roger','director':'Lewis'}, {'James':'Pierce', 'director':'Roger'}] k = 'James'

```
dlist = [{'James':'Sean', 'director':'Terence'}, {'James':'Roger','director':'Lewis'}, {'J
k = 'James'
print([i[k] for i in dlist])
```

```
['Sean', 'Roger', 'Pierce']
```

Task 0.5.22: Modify the comprehension in Task 0.5.21 to handle the case in which k might not appear in all the dictionaries. The comprehension evaluates to the list whose ith element is the value corresponding to key k in the ith dictionary in dlist if that dictionary contains that key, and 'NOT PRESENT' otherwise. Test your comprehension with k = 'Bilbo' and k = 'Frodo' and with the following list of dictionaries: dlist = [{'Bilbo':'Ian','Frodo':'Elijah'}, {'Bilbo':'Martin','Thorin':'Richard'}]

```
dlist = [{'Bilbo':'Ian','Frodo':'Elijah'}, {'Bilbo':'Martin','Thorin':'Richard'}]
k = 'Bilbo'
print([i[k] if k in i else 'NOT PRESENT' for i in dlist])
k = 'Frodo'
print([i[k] if k in i else 'NOT PRESENT' for i in dlist])
```

```
['Ian', 'Martin']
['Elijah', 'NOT PRESENT']
```

Task 0.5.23: Using range, write a comprehension whose value is a dictionary. The keys should be the integers from 0 to 99 and the value corresponding to a key should be the square of the key.

```
print({x:x**2 for x in range(100)})
```

```
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144
```

Task 0.5.24: Assign some set to the variable D, e.g. D ={'red','white','blue'}.Now write a comprehension that evaluates to a dictionary that represents the identity function on D.

```
D ={'red','white','blue'}
print({x+"Foo" for x in D})
```

```
{'redFoo', 'whiteFoo', 'blueFoo'}
```

Task 0.5.25: Using the variables base=10 and digits=set(range(base)), write a dictionary comprehension that maps each integer between zero and nine hundred ninety nine to the list of three digits that represents that integer in base 10. That is, the value should be {0: [0, 0, 0], 1: [0, 0, 1], 2: [0, 0, 2], 3: [0, 0, 3], ..., 10: [0, 1, 0], 11: [0, 1, 1], 12: [0, 1, 2], ..., 999: [9, 9, 9]}. Your expression should work for any base. For example, if you instead assign 2 to base and assign {0,1} to digits, the value should be {0: [0, 0, 0], 1: [0, 0, 1], 2: [0, 1, 0], 3: [0, 1, 1], ..., 7: [1, 1, 1]}.

```
base = 10
digits = set(range(base))
print({x:(x // base**2, x // base % base, x % base) for x in range(0, base**3)})
```

```
{0: (0, 0, 0), 1: (0, 0, 1), 2: (0, 0, 2), 3: (0, 0, 3), 4: (0, 0, 4), 5: (0, 0, 5), 6: (0,
```

Task 0.5.26: Suppose d is a dictionary that maps some employee IDs (a subset of the integers from 0 to n-1) to salaries. Suppose L is an n-element list whose ith element is the name of employee number i. Your goal is to write a comprehension whose value is a dictionary mapping employee names to salaries. You can assume that employee names are distinct. However, not every employee ID is represented in d. Test your comprehension with the following data: id2salary = {0:1000.0, 3:990, 1:1200.50} names = [‘Larry’, ‘Curly’, ’‘, ’Moe’]

```
id2salary = {0:1000.0, 3:990, 1:1200.50}
names = ['Larry', 'Curly', '', 'Moe']
print(names[i]:salary for (i, salary) in id2salary.items())
```

```
SyntaxError: invalid syntax (568612195.py, line 3)
```

Task 0.5.27: Try entering the definition of twice(z). After you enter the definition, you will see the ellipsis. Just press enter. Next, try invoking the procedure on some actual arguments. Just for fun, try strings or lists. Finally, verify that the variable z is now not bound to any value by asking {python} to evaluate the expression consisting of z.

```
def twice(z): return z+z
print(twice(2))
print(twice("foo"))
print(twice([1,2,3]))
print(z)
```

```
4
foofoo
[1, 2, 3, 1, 2, 3]
```

```
NameError: name 'z' is not defined
```

Task 0.5.28: Define a one-line procedure nextInts(L) specified as follows: input: list L of integers, output: list of integers whose ith element is one more than the ith element of L, example: input [1, 5, 7], output [2, 6, 8]

```
def nextInts(L): return [x+1 for x in L]
print(nextInts([1, 5, 7]))
```

```
[2, 6, 8]
```

Task 0.5.29: Define a one-line procedure cubes(L) specified as follows: input: list L of numbers, output: list of numbers whose ith element is the cube of the ith element of L, example: input [1, 2, 3], output [1, 8, 27].

```
def cubes(L): return [x**3 for x in L]
print(cubes([1, 2, 3]))
```

```
[1, 8, 27]
```

Task 0.5.30: Define a one-line procedure dict2list(dct,keylist) with this spec: input: dictionary dct, list keylist consisting of the keys of dct; output: list L such that L[i] = dct[keylist[i]] for i = 0, 1, 2,..., len(keylist)-1; example: input dct={'a':'A', 'b':'B', 'c':'C'} and keylist=['b','c','a']; output ['B', 'C', 'A']

```
def dict2list(dct,keylist): return [dct[x] for x in keylist]
print(dict2list({'a':'A', 'b':'B', 'c':'C'},['b','c','a']))
```

```
['B', 'C', 'A']
```

Task 0.5.31: Define a one-line procedure list2dict(L, keylist) specified as follows: input: list L, list keylist of immutable items; output: dictionary that maps keylist[i] to L[i] for i = 0, 1, 2,..., len(L)-1; example: input L=['A','B','C'] and keylist=['a','b','c']; output {'a':'A', 'b':'B', 'c':'C'}. Hint: Use a comprehension that iterates over a zip or a range.

```
def list2dict(L, keylist): return {keylist[i]:L[i] for i in range(len(L))}
print(list2dict(['A','B','C'],['a','b','c']))
```

```
{'a': 'A', 'b': 'B', 'c': 'C'}
```

Task 0.5.32: Write a procedure all_3_digit_numbers(base, digits) with the following spec: input: a positive integer base and the set digits which should be {0, 1, 2,..., base 1}; output: the set of all three-digit numbers where the base is base. For example, »> all_3_digit_numbers(2, {0,1}): {0, 1, 2, 3, 4, 5, 6, 7}; »> all_3_digit_numbers(3, {0,1,2}): {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26}; »> all_3_digit_numbers(10, {0,1,2,3,4,5,6,7,8,9}): {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, ... 985, 986, 987, 988, 989, 990, 991, 992, 993, 994, 995, 996, 997, 998, 999}.

```
def all_3_digit_numbers(base, digits): return {x*base**2+y*base+z for x in digits for y in
print(all_3_digit_numbers(2, {0,1}))
print(all_3_digit_numbers(3, {0,1,2}))
print(all_3_digit_numbers(10, {0,1,2,3,4,5,6,7,8,9}))
```

```
{0, 1, 2, 3, 4, 5, 6, 7}
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 2
{0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 2
```

Task 0.6.1: Import the math module using the command

```
import math
```

Task 0.6.2: The module random defines a procedure randint(a,b) that returns an integer chosen uniformly at random from among {a, a + 1,...,b}. Import this procedure using the command »> from random import randint. Try calling randint a few times. Then write a one-line procedure movie review(name) that takes as argument a string naming a movie, and returns a string review selected uniformly at random from among two or more alternatives (Suggestions: "See it!", "A gem!", "Ideological claptrap!").

```
from random import randint
def movie_review(name): return ["See it!", "A gem!", "Ideological claptrap!"][randint(0,2)
print(movie_review("movie"))
```

```
See it!
```

Task 0.6.3: In Tasks 0.5.30 and 0.5.31 of Lab 0.5, you wrote procedures dict2list(dct, keylist) and list2dict(L, keylist). Download the file dictutil.py from http://resources.codingthematrix.com. (That site hosts support code and sample data for the problems in this book.) Edit the provided file dictutil.py and edit it, replacing each occurence of pass with the appropriate statement. Import this module, and test the procedures. We will have occasion to use this module in the future.