Last name _____

First name _____

LARSON—OPER 635–SAGE WORKSHEET 11
Python Dictionaries

1. Log in to your Sage Cloud account.

   (a) Start Firefox or Chrome browser.

   (b) Go to `http://cloud.sagemath.com`

   (c) Click "Sign In".

   (d) Click project **Classroom Worksheets**.

   (e) Click "New", call it **s11**, then click "Sage Worksheet".

   Lists are ordered sets of objects, whereas **dictionaries** are unordered sets. But the main difference is that items in dictionaries are accessed via keys and not via their position. A dictionary is an associative array (also known as hashes). Any key of the dictionary is associated (or mapped) to a value. The values of a dictionary can be any Python data type. So dictionaries are unordered key-value-pairs.

   Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: . Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an *immutable* data type such as strings, numbers, or tuples.

2. Let's do an example. Evaluate: `dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}`. This defines a dictionary named *dict* with 3 keys: 'Name", 'Age' and 'Class', and 3 corresponding values.

3. Evaluate: `dict['Age']`.

4. To update a value associated to a key, simple reassign it. Evaluate: `dict['Age'] = 8`. Now evaluate: `dict['Age']` again.

5. To add a new entry to your dictionary, just add a new key and associated value. Let's add a new key 'School" and value "VCU". Evaluate: `dict['School'] = "VCU"`.

6. To see the current dictionary, evaluate: `dict`.

7. To remove an entry, use the `del` command. Let's remove the "Name" entry. Evaluate: `dict['Name']`. Now evaluate: `dict` to see the current dictionary.

8. To get the number of entries in a dictionary use the `len` function. Evaluate: `len(dict)`.

9. To get the dictionary entries as a list of key-value pairs, use the `items` method. Evaluate: `dict.items()`.

10. To get a list of the keys in a dictionary, use the `keys` methods. Evaluate: `dict.keys()`.

11. To get a list of the values in a dictionary, use the `values` methods. Evaluate: `dict.values()`.

12. You can iterate over the entries of your dictionary, analogous to how you do this for lists. Evaluate:

```
for key in dict:
    print dict[key]
```

Now the Sage LP solver produces the solution vector with variable labels as keys and numbers as the corresponding values. Let's revisit an example from the last worksheet.

13. Use the following code to define the graph (named g here) to Sage.

```
g=Graph(4)
for i in range(4):
    for j in range(4):
        g.add_edge((i,j),label=i+j)
```

14. Then use the following code to represent the *degree relaxation* TSP LP. Here we define a variable corresponding to each edge, and require that the sum of the decision variables corresponding to edges incident equal to any node be 2.

```
LP = MixedIntegerLinearProgram(maximization=False)
x = LP.new_variable(nonnegative=True)
LP.set_objective(sum([g.edge_label(i,j)*x[(i,j)] for i in range(4) \
    for j in range(4) if j>i]))
for v in g.vertices():
    LP.add_constraint(sum([x[u] for u in [(i,j) for i in range(4) \
        for j in range(4) if (i==v and j>i) or (j==v and i<j)]]) == 2)
for e in g.edges(labels=False):
    LP.add_constraint(x[e], max=1)
```

15. To solve this LP evaluate, `LP.solve()`. To get the corresponding values for the decision variables evaluate `LP.get_values(x)`. This actually produced a Python dictionary. Let's give it the name D. Evaluate: `D=LP.get_values(x)`.

16. Evaluate: `D`.

17. Evaluate: `D.keys()`.

18. Evaluate: `D.values()`.

19. Evaluate: `D.items()`.

20. To print key-value pairs only in the case where the value is positive, evaluate:

```
for key in D:
    if D[key] > 0:
        print key, D[key]
```