Module 1 Activity

Assigned at the start of Module 1

Due at the end of Module 1

Weekly Discussion Forum Participation

Each week, you are required to participate in the module's discussion forum. The discussion forum consists of the week's Module Activity, which is released at the beginning of the module. You must complete/attempt the activity before you can post about the activity and anything that relates to the topic.

Grading of the Discussion

1. Initial Post:

Create your thread by **Day 5 (Saturday night at midnight, PST).**

2. Responses:

Respond to at least two other posts by **Day 7 (Monday night at midnight, PST).**

Grading Criteria:

Your participation will be graded as follows:

Full Credit (100 points):

- Submit your initial post by **Day 5.**
- Respond to at least two other posts by **Day 7.**

Half Credit (50 points):

- If your initial post is late but you respond to two other posts.
- If your initial post is on time but you fail to respond to at least two other posts.

No Credit (0 points):

- If both your initial post and responses are late.
- If you fail to submit an initial post and do not respond to any others.

Additional Notes:

- **Late Initial Posts:** Late posts will automatically receive half credit if two responses are completed on time.
- **Substance Matters:** Responses must be thoughtful and constructive. Comments like "Great post!" or "I agree!" without further explanation will not earn credit.
- **Balance Participation:** Aim to engage with threads that have fewer or no responses to ensure a balanced discussion.

Avoid:

- A number of posts within a very short time-frame, especially immediately prior to the posting deadline.
- Posts that complement another post, and then consist of a summary of that.

```
import pandas as pd
from sklearn.datasets import load_iris

iris = load_iris()

iris_df = pd.DataFrame(
    data=iris.data,
    columns=iris.feature_names
)
iris_df['target'] = iris.target

iris_df.head()
```

```
   sepal length (cm)  sepal width (cm)  petal length (cm)  petal width
(cm)  \
0                5.1               3.5                1.4
0.2
1                4.9               3.0                1.4
0.2
2                4.7               3.2                1.3
0.2
3                4.6               3.1                1.5
0.2
4                5.0               3.6                1.4
0.2

   target
0       0
1       0
2       0
3       0
4       0
```

1. Module 1

Python has four inbuilt data structures as follows:

1. Lists
2. Dictionary
3. Tuple
4. Set

For each of these data structures use the Iris data set and provide the following:

(a) Lists

- Define a "list" as comprehensively and accurately as you would find in a dictionary. Provide a clear and concise explanation, covering its essential meaning, characteristics, and any relevant context.
- Create a list in Python using the first observation, all features and label from the Iris data set
- Show the output with a print statement
- Add the second observation of the Iris data set to your list
- Show the output of your list with a print statement

List is an ordered and mutable collection of items to stores a sequence of elements that can be of different types (e.g., numbers, strings, objects, or even other lists)

```
## import libraries
import pandas as pd

## import iris dataset
from sklearn.datasets import load_iris

iris = load_iris()

## Type code here for the problem ##

# Load as DataFrame
iris_df = pd.DataFrame(
    data = iris.data,
    columns = iris.feature_names
)
# Add target as additional column to data frame
iris_df['target'] = iris.target

# Get the first observation i.e. first row, all features i.e. all the
columns and label i.e. add target column as well
```

```
first_observation_series = iris_df.iloc[0]

# Convert panda series into python list
observation_list = [first_observation_series.tolist()]

# Print the first observation as a python list
print("First row observation list")
print(observation_list)

# Read second observation
second_observation_series = iris_df.iloc[1]
second_observation_list = second_observation_series.tolist()

# Apend second observation list to observation list
observation_list.append(second_observation_list)

# Print after adding the second observation
print("After adding the second row observation")
print(observation_list)
```

```
First row observation list
[[5.1, 3.5, 1.4, 0.2, 0.0]]
After adding the second row observation
[[5.1, 3.5, 1.4, 0.2, 0.0], [4.9, 3.0, 1.4, 0.2, 0.0]]
```

(b) Dictionary

- Define a "dictionary" as comprehensively and accurately as you would find in a dictionary. Provide a clear and concise explanation, covering its essential meaning, characteristics, and any relevant context.
- Create a dictionary in Python using the classes of Setosa, Versicolor and Virginica, allow elements (2 observations per class) to be added
- Show the output with a print statement
- Add a third observation of the Iris data set for each class
- Show the output of your with a print statement

Dictionary is an unordered and mutable collection of key/value pair with faster read/write/delete operation based on hashable key. It is used to define structured data.

```
## Type code here for the problem ##
# Read the species names
species = [str(target) for target in iris.target_names]

# Group DataFrame as per species name
species_df_dict = {species_name: iris_df[iris_df['target'] ==
speceis_id]  for speceis_id, species_name in enumerate(species) }
```

```
# Dictionary of species with two observations
species_dict_observations = {species_name:
species_df_dict[species_name].iloc[0:2].values.tolist() for
species_name in species }
print("Flower classes with two observations")
print(species_dict_observations)

# Add one more observation with each species
for species_name in species:

species_dict_observations[species_name].append(species_df_dict[species
_name].iloc[2].tolist())
print("Flower classes with three observations")
print(species_dict_observations)
```

```
Flower classes with two observations
{'setosa': [[5.1, 3.5, 1.4, 0.2, 0.0], [4.9, 3.0, 1.4, 0.2, 0.0]],
'versicolor': [[7.0, 3.2, 4.7, 1.4, 1.0], [6.4, 3.2, 4.5, 1.5, 1.0]],
'virginica': [[6.3, 3.3, 6.0, 2.5, 2.0], [5.8, 2.7, 5.1, 1.9, 2.0]]}
Flower classes with three observations
{'setosa': [[5.1, 3.5, 1.4, 0.2, 0.0], [4.9, 3.0, 1.4, 0.2, 0.0],
[4.7, 3.2, 1.3, 0.2, 0.0]], 'versicolor': [[7.0, 3.2, 4.7, 1.4, 1.0],
[6.4, 3.2, 4.5, 1.5, 1.0], [6.9, 3.1, 4.9, 1.5, 1.0]], 'virginica':
[[6.3, 3.3, 6.0, 2.5, 2.0], [5.8, 2.7, 5.1, 1.9, 2.0], [7.1, 3.0, 5.9,
2.1, 2.0]]}
```

(c) Tuple

- Define a "tuple"" as comprehensively and accurately as you would find in a dictionary. Provide a clear and concise explanation, covering its essential meaning, characteristics, and any relevant context.
- Create a tuple in Python using the classes of Setosa, Versicolor and Virginica, allow elements (start with 2 observations and 2 features per class) to be added
- Show the output with a print statement
- Add a third observation and a third feature of the Iris data set for each class
- Show the output of your with a print statement

Tuple is similar to lists, but immutable i.e. ordered and immutable collection of items to stores a sequence of elements that can be of different types.

```
## Type code for the problem here ##
# Get species names
species = [str(species_name) for species_name in iris.target_names]
```

```
# Two features to be included
features_to_include = iris.feature_names[0:2]

# Filter DataFrame with included features as well as target
all_columns = features_to_include + ['target']
subset_df = iris_df[all_columns]

# Group Dataframe as per species
species_dict_df = { species_name: subset_df[subset_df['target'] ==
species_id]  for species_id, species_name in enumerate(species)}

# Build list with two rows
species_observations_list = [species_dict_df[species_name]
[0:2].values.tolist() for species_name in species]
observations_tuples= tuple(tuple(tuple(observation) for observation in
species_observations) for species_observations in
species_observations_list)

print("Tuple with two observations with two features for each
classes")
print(observations_tuples)

# Add third observation with third feature to tuple
# Since tuple are immutable therefore we can't modify
observations_tuples, we will have to create a new tuple
features_to_include = iris.feature_names[0:3]
all_columns = features_to_include + ['target']
subset_df = iris_df[all_columns]
species_dict_df = { species_name: subset_df[subset_df['target'] ==
species_id]  for species_id, species_name in enumerate(species)}
species_observations_list = [species_dict_df[species_name]
[0:3].values.tolist() for species_name in species]
observations_tuples= tuple(tuple(tuple(observation) for observation in
species_observations) for species_observations in
species_observations_list)
print("Tuple with three observations with three features for each
classes")
print(observations_tuples)
```

```
Tuple with two observations with two features for each classes
(((5.1, 3.5, 0.0), (4.9, 3.0, 0.0)), ((7.0, 3.2, 1.0), (6.4, 3.2,
1.0)), ((6.3, 3.3, 2.0), (5.8, 2.7, 2.0)))
Tuple with three observations with three features for each classes
(((5.1, 3.5, 1.4, 0.0), (4.9, 3.0, 1.4, 0.0), (4.7, 3.2, 1.3, 0.0)),
((7.0, 3.2, 4.7, 1.0), (6.4, 3.2, 4.5, 1.0), (6.9, 3.1, 4.9, 1.0)),
((6.3, 3.3, 6.0, 2.0), (5.8, 2.7, 5.1, 2.0), (7.1, 3.0, 5.9, 2.0)))
```

(d) Set

- Define a "set"" as comprehensively and accurately as you would find in a dictionary. Provide a clear and concise explanation, covering its essential meaning, characteristics, and any relevant context.
- Create a set (1st set) in Python using the first observation of the Setosa class allowing elements to be added
- Show the output with a print statement
- Create another set (2nd set) by adding the second observation of the Setosa class
- Show the output of the following operations with the 1st set and 2nd set with a print statement
  - Difference
  - Intersection
  - Union

Set is unordered collection of unique elements. It is mutable and element must be hashable

```
## Type the code for the problem here ##
# Filter Dataframe for Setosa class is with index 0
setosa_df=iris_df[iris_df['target'] == 0]

# Read first observation of setosa
setosa_one_observation = set(setosa_df.iloc[0].tolist())
print("Set of Setosa one observation")
print(setosa_one_observation)

# Create copy to build Setosa with two observations
setosa_two_observation = setosa_one_observation.copy()
for x in setosa_df.iloc[1]:
    setosa_two_observation.add(x)
print("Set of Setosa two observations")
print(setosa_two_observation)

# Perform set difference operation
difference=setosa_two_observation.difference(setosa_one_observation)
print("Difference operation on set")
print(difference)

# Perform set intersection operation
intersection=setosa_two_observation.intersection(setosa_one_observatio
n)
print("Intersection operation on set")
print(intersection)

# Perform set union operation
union=setosa_two_observation.union(setosa_one_observation)
```

```
print("Union operation on set")
print(union)
```

```
Set of Setosa one observation
{0.0, 3.5, 5.1, 1.4, 0.2}
Set of Setosa two observations
{0.0, 5.1, 3.5, 1.4, 3.0, 4.9, 0.2}
Difference operation on set
{4.9, 3.0}
Intersection operation on set
{0.0, 3.5, 5.1, 1.4, 0.2}
Union operation on set
{0.0, 3.5, 3.0, 5.1, 1.4, 4.9, 0.2}
```

References

[1] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms, Third Edition. MIT Press and McGraw-Hill, 2009. ISBN-13: 978-0-262-03384-8 [2] Thomas H. Cormen. The clrscode and clrscode3e packages for LaTeX2e, Retrived Jan 2010, http://www.cs.dartmouth.edu/ thc/clrscode/ [3] simpleilearn, Top 90+ Data Science Interview Questions and Answers: Basic to Technical, https://www.simplilearn.com/tutorials/data-science-tutorial/data-science-interview-questions [4] simpleilearn, 22 Artificial Intelligence Interview Questions to Prepare, https://www.simplilearn.com/artificial-intelligence-ai-interview-questions-and-answers-article [5] Python, Data Structures, https://docs.python.org/3/tutorial/datastructures.html