Practice Questions for Final

Here are some representative practice questions for the Final Exam on April 19th. Remember this will be a take-home open-book exam so you can look things up in your lecture notes and practical assignments (for example).

The April 1st Tuesday practical session will provide an opportunity for you to attempt these questions and the April 3rd Thursday practical will involve a recitation where the TAs will go over the solutions.

Your solutions do not need to be submitted and will not be graded.

Q1 [2 points]. What will print(Y.x) output and why?

```
class X(object):
    x = 1
    y = 10
    def __init__(self):
        self.instance_x = 100
    def get_x(self):
        return self.x

class Y(X):
    z = 20
    def __init__(self):
        super().__init__()
        self.instance_y = 200

class Z(X):
    w = 30
    def __init__(self):
        super().__init__()
        self.instance_x = 300

X.x = 2
Z.x = 3
```

Q2 [2 points] Describe 1 of the general differences in approach between Machine Learning and Statistics?

Q3 [4 points] Identify and fix the bug in this code designed to simulate chemical collisions?

```
import numpy as np

def simulate_reaction_kinetics(num_molecules=1000, temperature=300,
simulation_time=10.0, seed=42):
       """
```

```
        Simulate molecular collisions in a chemical reaction.

        Parameters:
        num_molecules (int): Initial number of molecules
        temperature (float): Temperature in Kelvin
        simulation_time (float): Total simulation time in seconds
       seed (integer): Random seed to use

        Returns:
        tuple: (collision_times, remaining_molecules)
        """
       # Constants
        collision_rate = 0.5 * (temperature / 300)  # collisions per molecule per
second

       # Set up PRNG
       rng = np.random.default_rng(seed)

        # Arrays to store results
        collision_times = []
        remaining_molecules = [num_molecules]
        current_time = 0.0

        # Simulate until we run out of time or molecules
        while current_time < simulation_time and remaining_molecules[-1] > 0:
               # Calculate mean time to next collision for remaining molecules
               mean_time = 1.0 / (collision_rate * remaining_molecules[-1])

               # Simulate time until next collision with exponential distribution
               next_collision_time = rng.random() * mean_time * 2

               # Update the current time
               current_time += next_collision_time

               if current_time < simulation_time:
                      # Record the collision
                      collision_times.append(current_time)

                      # One molecule reacts and is removed
                      remaining_molecules.append(remaining_molecules[-1] - 1)

        return np.array(collision_times), np.array(remaining_molecules)

# Run the simulation
collision_times, remaining_molecules = simulate_reaction_kinetics()
```

Q4 [2 points] What is c?

```
a = [[3,5],[7,11]]
b = a
c = a[:]
a[0][1] = 4
c[1] = b[0]
```

a) [[3,5],[7,11]]

b) [[3,5],[3,5]]

c) [[3,4],[3,5]]

d) [[3,4],[3,4]]

Q5 [2 points] What is y?

```
y = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19]
y = y[3:15:3][1:4:2]
```

a) [3,6,9,12,15]

b) [7,13]

c) [1,9]

d) [4,7,10,13,2,4]

e) TypeError

Q6 [2 points]: You want to plot the function y=2x. Identify and fix the bug in this code?

```
import matplotlib.pyplot as plt
x=[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
plt.plot(x, 2*x)
```

Q7 [2 points]What does `"abc" and 42` evaluate to? Why?

a) False

b) True

c) "abc"

d) 42

e) TypeError

Q8 [4 points] Explain how simulation is used as a way to calculate p-values?

Q9 [2 points] What will `str(float(int(float("7.5"))))` evaluate to:

a) 7

b) 7.0

c) 7.5

d) "7"

e) "7.0"

f) "7.5"

Q10 [4 points] Explain the concept of inheritance in object-oriented programming in python and provide an example of when you might use it?

Q11 [2 points] Fix this function so it returns strings with "_" a

```
def add_underscores(word):
    """
    This function adds underscores between each letter
    of the input word and returns the new word
    Input:
       word(str) : string containing characters
    Output:
       new_word(str): input word string with "_" characters added between
                      each pair of letters
    """
    new_word = "_"
    for char in word:
        new_word = char + "_"
    return new_word

phrase = "hello"
add_underscores(phrase)
```

Q12 [2 points] Why might the following code cause an error? What alternative approach could you take in python that would avoid this error?

```
empty_list = [0] * (10**9999)
```

Q13 [2 points]. What does this print?

```
class A:
    name = "Default"
    def __init__(self, s):
        self.name = s

a = A('Bob')
print(a.name)
print(A.name)
```

Q14 [4 points] Explain the difference between mutable and immutable data types in python and provide an example of each.

Q15 [2 points]. What will this code print?

```
print([3 * x for x in range(5) if x % 5 == 0])
```

Q16 [4 points]. What will `contents` contain after we run this code?

```
with open("test.txt", "w") as f:
    f.write("Hello\nWorld")

with open("test.txt", "w") as f:
    f.write("Yes\n")

with open("test.txt", "a") as f:
    f.write("End\nExam")

with open("test.txt", "r") as f:
    contents = f.readline()
```

Q17 [4 points] Provide 3 methods by which you can fit a straight line to a dataset?

Q18 [4 points] Fix the bug in this recursive function for generating the Fibonacci sequence

```
def fibonacci(n):
    """
    Calculate the nth Fibonacci number using recursion.
    The Fibonacci sequence starts with 0, 1, 1, 2, 3, 5, 8, 13, ...

    Parameters:
    n (int): The position in the Fibonacci sequence (0-indexed)

    Returns:
    int: The nth Fibonacci number
    """
    # Base cases
    if n <= 0:
        return 0
    if n == 1:
        return 1

    # Recursive case: sum of the two previous Fibonacci numbers
    return fibonacci(n) - fibonacci(n-1)

# Test the function
```

```
for i in range(10):
    print(f"Fibonacci({i}) = {fibonacci(i)}")
```

Q19 [4 points] Rewrite the following code using A) list comprehension and B) the filter function

```
temperatures = [22.5, 19.8, -5.0, 35.2, 102.5, 23.1, -15.7, 27.4]
filtered_temps = []
for temp in temperatures:
    if temp <= 0.0:
        filtered_temps.append(temp)
```

Q20 [3 points]. What is the value of x, y, and z after this code finishes?

```
x = {'a': 1, 'b': 2}
y = x
z = x.copy()
y['c'] = 3
z['a'] = 10
```

Q21 [2 points]. What will be stored in `result`?

```
def combine_dicts(d1, d2):
    result = d1.copy()
    for key, value in d2.items():
        if key in result:
            result[key] = [result[key], value]
        else:
            result[key] = value
    return result

result = combine_dicts({'a': 1, 'b': 2}, {'b': 3, 'c': 4})
```

Q22 [2 points]. What will this comprehension evaluate to?

```
{x: x**2 for x in range(20) if x % 4 == 0}
```

Q23 [2 points]. What will the result variable contain after this code has been run?

```
import pandas as pd
import numpy as np
df = pd.DataFrame({'A': [1, 2, 3], 'B': [4, 5, 6]})
result = df.mean(axis=0)
```

Q24 [2 points]. What will be the shape of the result variable after this code runs?

```
import pandas as pd
df = pd.DataFrame({'A': [1, 2, 3, 4], 'B': [5, 6, 7, 8]})
result = df.drop(index=1, axis=0)
```

Q25 [3 points]. Describe 3 different file formats that Pandas allows users to read directly into DataFrames.

Q26 [2 points]. What will `x` contain?

```
import numpy as np
arr = np.array([1, 2, 3, 4, 5])
x = arr[1:4]
```

Q28 [2 points]. What does this print?

```
import numpy as np
a = np.array([1, 2, 3])
b = np.array([4, 5, 6])
print(np.concatenate((a, b)))
```

Q29 [2 points]. Explain the difference between a Pandas Series and a Pandas DataFrame.

Q30 [2 points] What will this plot contain?

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 2*np.pi, 100)
plt.plot(x, np.sin(x))
plt.plot(x, np.cos(x))
plt.show()
```