Name of Candidate:

Use Python 3. You may use anything from Python's standard library.

**## Question 1**

Why does the following program print `prime` for input `6`? How should

it be modified to print only when the value is one of the first four

primes?

```python

x = 6

if x == 2 or 3 or 5 or 7:

print('prime')

```

**### Answer 1**

*The equality looks at the conditions like (x==2) or (3) or (5) or (7). Although x==2 is not true when x = 6, the next equality check is just the number 3, which is a truthy, therefore ‘prime’ is printed.*

**## Question 2**

Which of these is the most efficient, and why?

```python

x in set([2, 3, 5, 7])

x in list([2, 3, 5, 7])

x in tuple([2, 3, 5, 7])

**### Answer 2**

*The set is the most efficient since under the hood it is a hash table. The list and tuple take longer to search because it would be linear.*

**## Question 3**

The following code should print `[5], [5]`. Why doesn't it produce

the expected output? How should it be fixed?

```python

def init\_data(data=[]):

"""Create a list containing the value 5.""".

data.append(5)

return data

x = init\_data()

y = init\_data()

print(x, y)

```

**### Answer 3**

*The empty list that is the default value for data is initialize when the function is initialized, therefore x and y shared the same list.*

**## Question 4**

What does the following code do? What is the difference between the two

versions?

```python

if any([value % 2 for value in values]):

print('done')

# or

if any(value % 2 for value in values):

print('done')

```

**### Answer 4**

*The above checks if any value is truthy in a list while below checks it in a generator object. Both will print ‘done’ because both have truthy values in them.*

**## Question 5**

What is the equivalent of the following code without using decorator

syntax? Does order matter when decorating a function multiple times?

```python

@bp.route('/info')

@login\_required

def user\_info():

return jsonify(current\_user)

```

**### Answer 5**

*def user\_info():*

*return jsonify(current\_user)*

*user\_info = login\_required(user\_info)*

*user\_info = bp.route('/info')(user\_info)*

*Yes, the order matters when decorating a function multiple times.*

*The decorators are applied from the bottom up. In this case, `login\_required`*

*is applied first, then `bp.route('/info')`.*

**## Question 6**

A list contains instances of the following class. Write a function that

will produce a sorted list ordered first by name descending, then id

ascending.

```python

class User:

def \_\_init\_\_(self, id, name):

self.id = id

self.name = name

def sort\_name\_desc\_id\_asc(users):

# complete this function

pass

users = [User(...), User(...), ...]

sorted\_users = sort\_name\_desc\_id\_asc(users)

```

**### Answer 6**

*def sort\_name\_desc\_id\_asc(users:list[User]):*

*return sorted(users, key=lambda x: (-ord(x.name[0]), x.id))*

**## Bonus**

What does the following output, and why?

```python

x = [1, 2, 3, 4]

for x[-1] in x:

print(x)

```

**### Bonus Answer**

*The output is*

*[1, 2, 3, 1]*

*[1, 2, 3, 2]*

*[1, 2, 3, 3]*

*[1, 2, 3, 3]*

*This is because the first iteration assigns 1 to index -1. The second iteration assigns 2 to index -1. The third assigns 3 to index -1. The fourth iteration assigns 3 (because of the previous iteration) to index -1.*