List Manipulation

LATEST SUBMISSION GRADE

100%

Try again

1. Given the list `my_list = [1, 3, 5, 7, 9]`, which of the following slices returns the list `[3, 5, 7, 9]`? 1 / 1 point

Grade 100% — View Feedback — We keep your highest score

- [ ] `my_list[ 2 : 4]`
- [x] `my_list[ 1 : ]`

✓ **Correct**

This slice returns the list `[3, 5, 7, 9]`.

- [ ] `my_list[ 1 : 4]`
- [ ] `my_list[ 1 : -1]`

2. While of the following expressions returns a tuple of length one? 1 / 1 point

- [x] `(1,)`

✓ **Correct**

This expression returns the tuple `(1, )`.

- [ ] `(1)`
- [ ] `[1]`
- [x] `tuple([1])`

✓ **Correct**

This expression returns the tuple `(1, )`.

3. Why does following code snippet raise an error in Python? 1 / 1 point

```
1  instructors = ("Scott", "Joe", "John", "Stephen")
2  instructors[2 : 4] = []
3  print(instructors)
```

- The tuple doesn't contain an element with index 4.
- Slices cannot be used with tuples.
- John and Stephen are irreplaceable.
- (•) Tuples are immutable.

✓ **Correct**

4. Given a non-empty list `my_list`, which item in the list does the operation `my_list.pop()` remove? 1 / 1 point

- The item `my_list[len(my_list) ]`
- (•) The item `my_list[-1]`
- The item `my_list[0]`
- The item `my_list[1]`

✓ **Correct**

The method `pop()` removes the last item in the list.

5. What output does the following code snippet print to the console? 1 / 1 point

```
1  my_list = [1, 3, 5, 7, 9]
2  my_list.reverse()
3  print(my_list.reverse())
```

Note that this question is easily answered by running this snippet in Python. Instead, carefully evaluate this code snippet mentally when you attempt this problem.

- `[9, 7, 5, 3, 1]`
- `[1,  3, 5, 7, 9]`
- Executing this code snippet raises an error.
- (•) `None`

✓ **Correct**

Since `reverse()` is a method, it mutates `my_list` and returns `None`.

6. Given a list `fib = [0, 1]`, write a loop that appends the sum of the last two items in `fib` to the end of `fib`. What is the value of the last item in `fib` after twenty iterations of this loop? Enter the answer below as an integer. 1 / 1 point

As a check, the value of the last item in `fib` after ten iterations is 89.

10946

✓ **Correct**

Correct. The values in this list are the Fibonacci numbers.

7. One of the first examples of an algorithm was the Sieve of Eratosthenes. This algorithm computes all prime numbers up to a specified bound. The provided code below implements all but the innermost loop for this algorithm in Python. Review the linked Wikipedia page and complete this code. 1 / 1 point

```
1  """
2  Implement the Sieve of Eratosthenes
3  https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
4  """
5
6  def compute_primes(bound):
7      """
8      Return a list of the prime numbers in range(2, bound)
9      """
10
11     answer = list(range(2, bound))
12     for divisor in range(2, bound):
13         # Remove appropriate multiples of divisor from answer
14         pass
15     return answer
16
17 print(len(compute_primes(200)))
18 print(len(compute_primes(2000)))
```

Running your completed code should print two numbers in the console. The first number should be 46. Enter the second number printed in the console as the answer below.

303

✓ **Correct**

There are exactly 303 prime numbers less than 2000. Here is our solution:

```
1  """
2  Implement the Sieve of Eratosthenes
3  https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes
4  """
5
6  def compute_primes(bound):
7      """
8      Return a list of the prime numbers in range(2, bound)
9      """
10
11     answer = list(range(2, bound))
12     for divisor in range(2, bound):
13         for stride in range(2 * divisor, bound, divisor):
14             if stride in answer:
15                 answer.remove(stride)
16     return answer
17
18 print(len(compute_primes(200)))
19 print(len(compute_primes(2000)))
```

Note that our code is not particularly efficient and can be optimized in several ways.