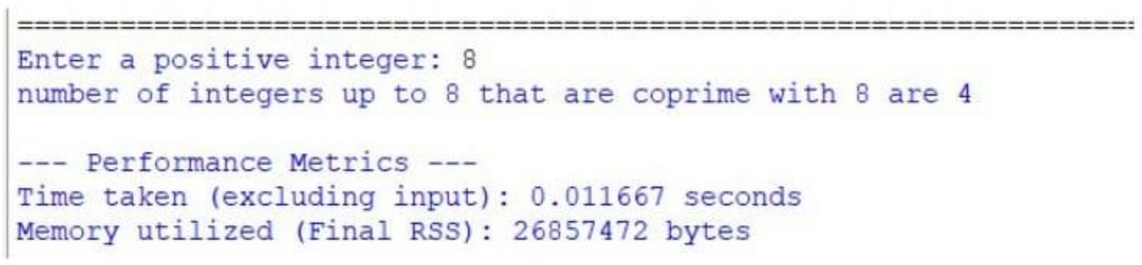
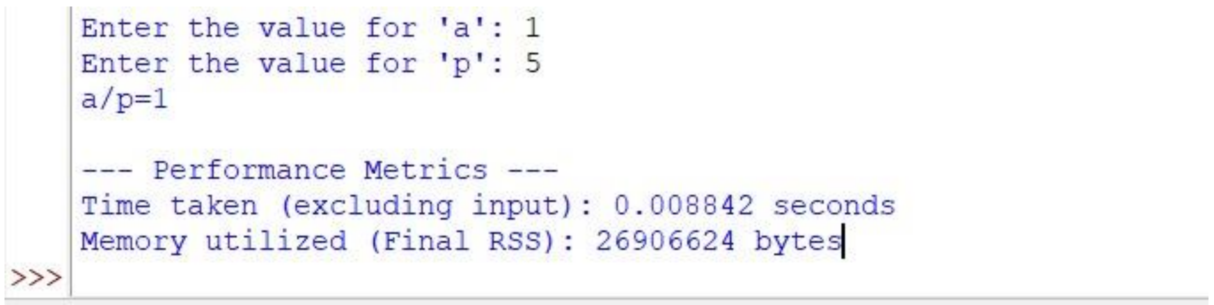
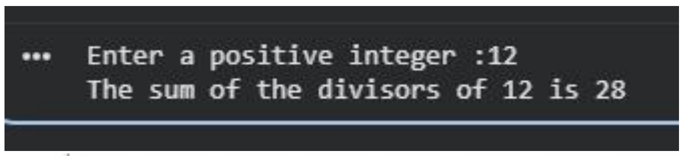


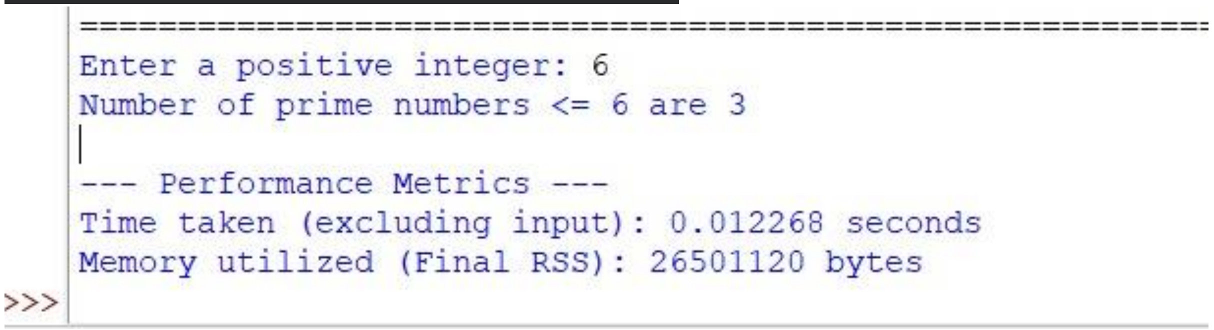
```
==========================================================================
Enter a positive integer: 8
number of integers up to 8 that are coprime with 8 are 4

--- Performance Metrics ---
Time taken (excluding input): 0.011667 seconds
Memory utilized (Final RSS): 26857472 bytes
```



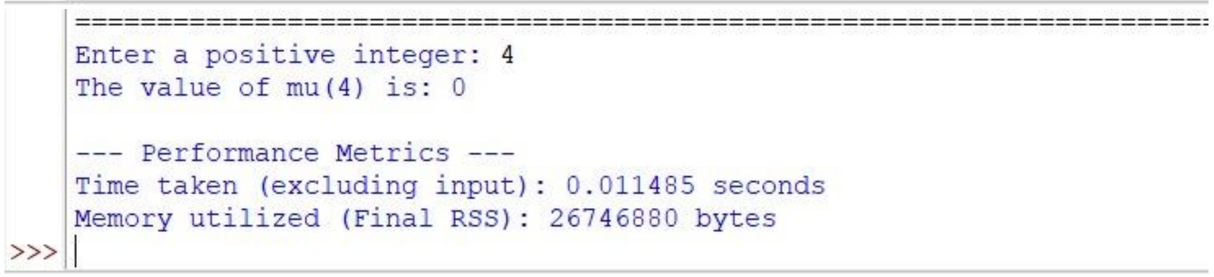
```
Enter the value for 'a': 1
Enter the value for 'p': 5
a/p=1

--- Performance Metrics ---
Time taken (excluding input): 0.008842 seconds
Memory utilized (Final RSS): 26906624 bytes
>>>
```



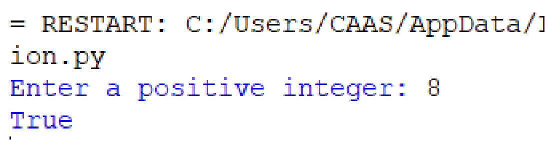
```
Enter a positive integer :12
The sum of the divisors of 12 is 28
```



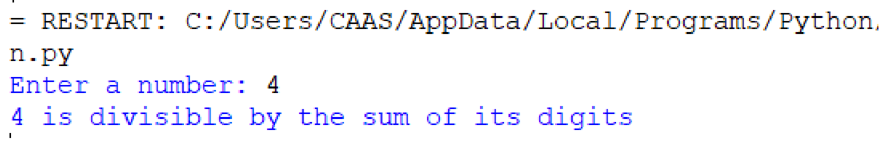
```
==========================================================
Enter a positive integer: 6
Number of prime numbers <= 6 are 3

--- Performance Metrics ---
Time taken (excluding input): 0.012268 seconds
Memory utilized (Final RSS): 26501120 bytes
>>>
```



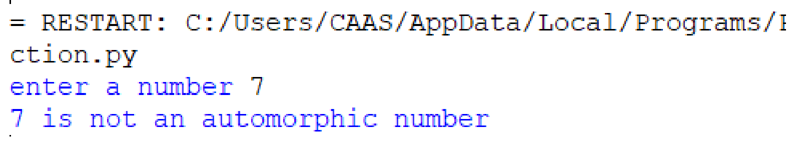
```
==========================================================================
Enter a positive integer: 4
The value of mu(4) is: 0

--- Performance Metrics ---
Time taken (excluding input): 0.011485 seconds
Memory utilized (Final RSS): 26746880 bytes
>>>
```



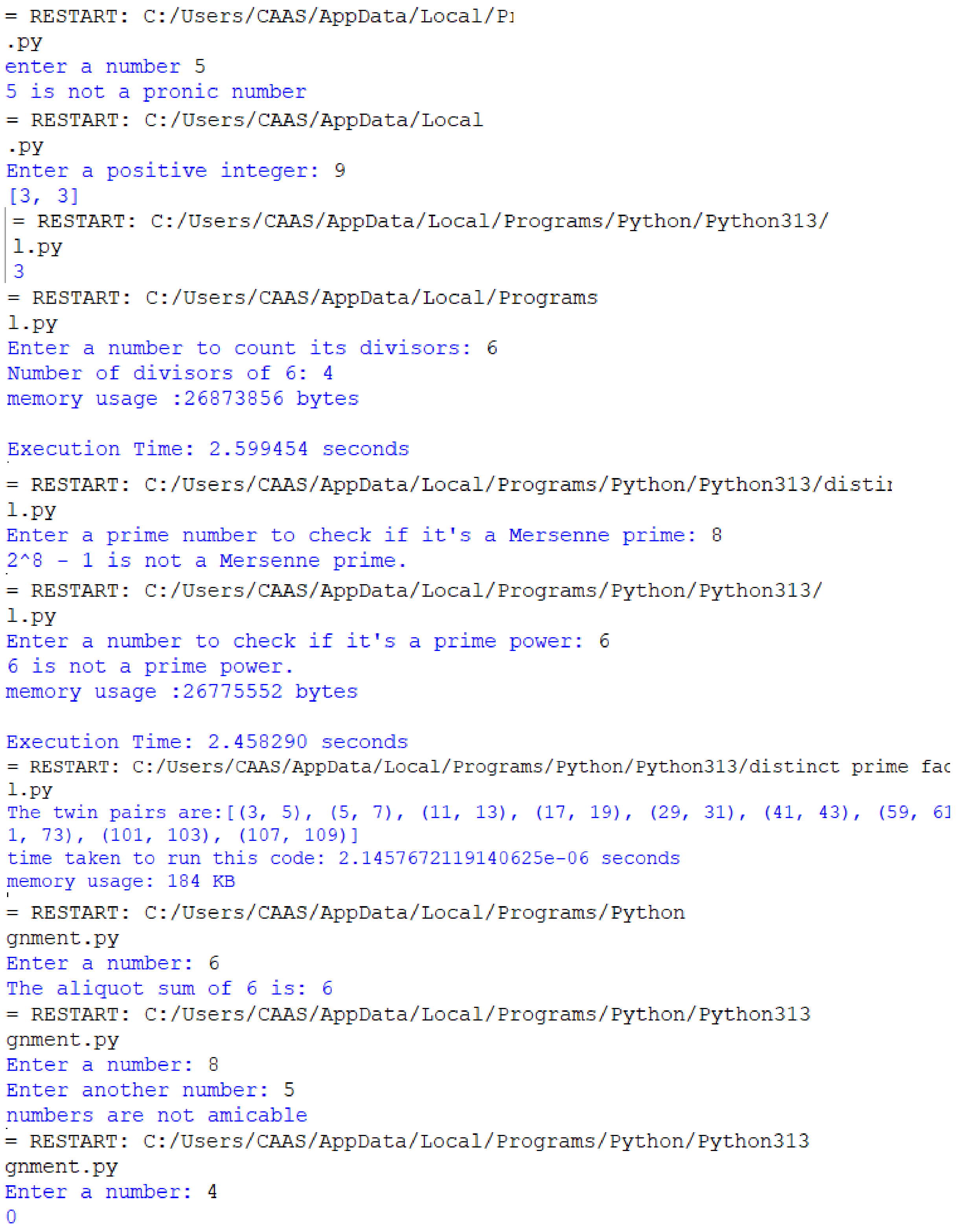
```
= RESTART: C:/Users/CAAS/AppData/
ion.py
Enter a positive integer: 8
True
```



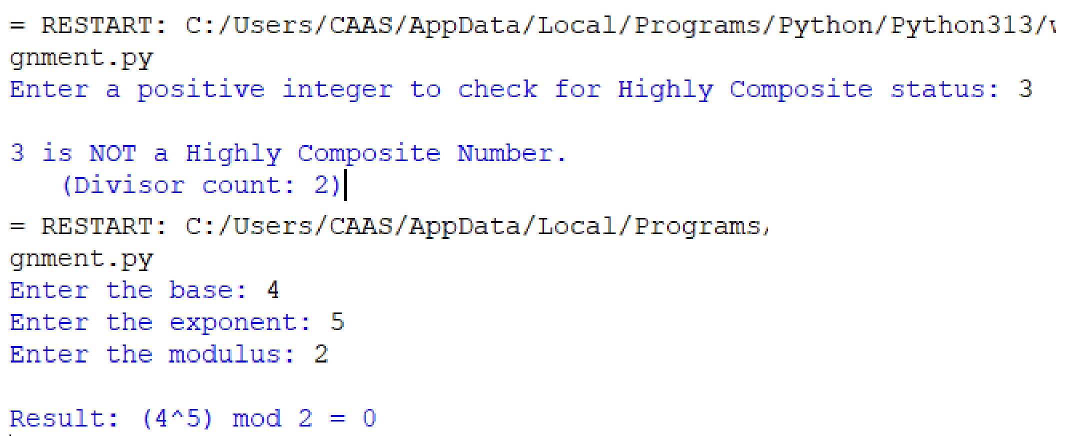
```
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python
n.py
Enter a number: 4
4 is divisible by the sum of its digits
```



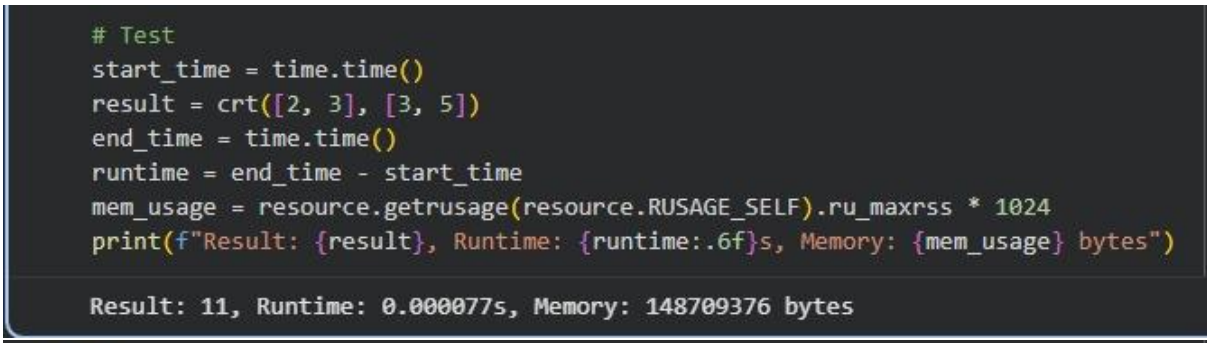
```
= RESTART: C:/Users/CAAS/AppData/Local/Programs/
ction.py
enter a number 7
7 is not an automorphic number
```



```
= RESTART: C:/Users/CAAS/AppData/Local/P
.py
enter a number 5
5 is not a pronic number
= RESTART: C:/Users/CAAS/AppData/Local
.py
Enter a positive integer: 9
[3, 3]
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313/
l.py
3
= RESTART: C:/Users/CAAS/AppData/Local/Programs
l.py
Enter a number to count its divisors: 6
Number of divisors of 6: 4
memory usage :26873856 bytes

Execution Time: 2.599454 seconds
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313/disti
l.py
Enter a prime number to check if it's a Mersenne prime: 8
2^8 - 1 is not a Mersenne prime.
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313/
l.py
Enter a number to check if it's a prime power: 6
6 is not a prime power.
memory usage :26775552 bytes

Execution Time: 2.458290 seconds
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313/distinct prime fac
l.py
The twin pairs are:[(3, 5), (5, 7), (11, 13), (17, 19), (29, 31), (41, 43), (59, 61
1, 73), (101, 103), (107, 109)]
time taken to run this code: 2.1457672119140625e-06 seconds
memory usage: 184 KB
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python
gnment.py
Enter a number: 6
The aliquot sum of 6 is: 6
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313
gnment.py
Enter a number: 8
Enter another number: 5
numbers are not amicable
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313
gnment.py
Enter a number: 4
0
```

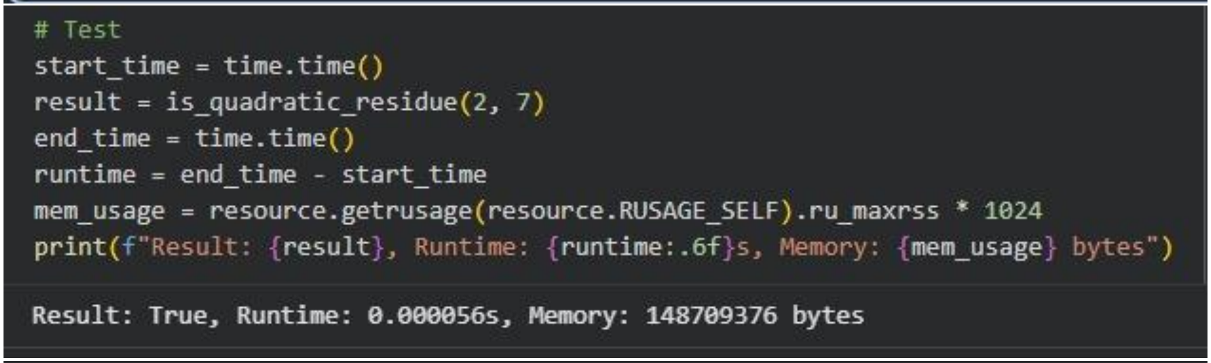


```
= RESTART: C:/Users/CAAS/AppData/Local/Programs/Python/Python313/\
gnment.py
Enter a positive integer to check for Highly Composite status: 3

3 is NOT a Highly Composite Number.
   (Divisor count: 2)
= RESTART: C:/Users/CAAS/AppData/Local/Programs/
gnment.py
Enter the base: 4
Enter the exponent: 5
Enter the modulus: 2

Result: (4^5) mod 2 = 0
```



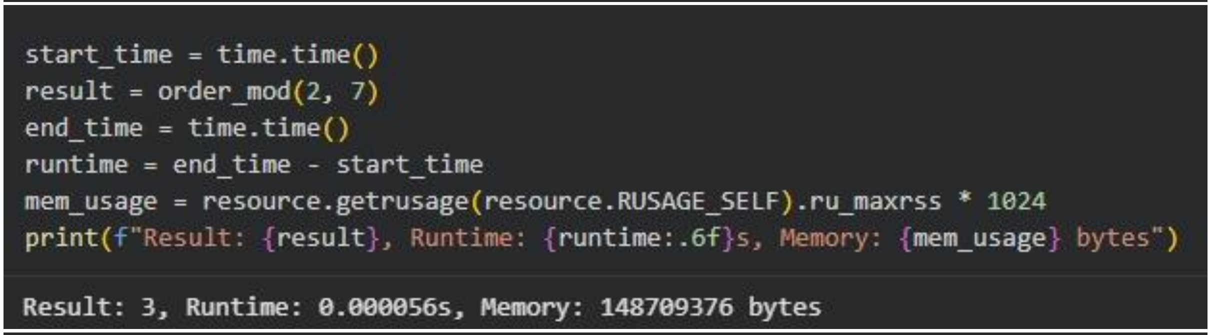
```
# Test
start_time = time.time()
result = crt([2, 3], [3, 5])
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 11, Runtime: 0.000077s, Memory: 148709376 bytes
```



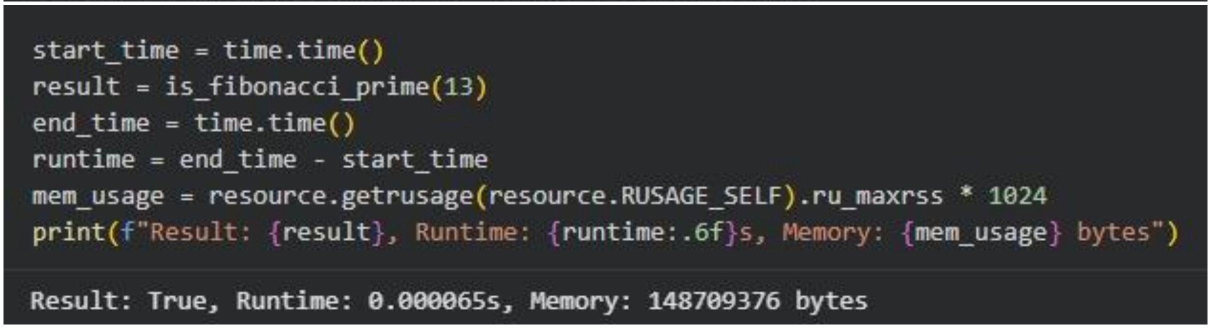
```
# Test
start_time = time.time()
result = is_quadratic_residue(2, 7)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: True, Runtime: 0.000056s, Memory: 148709376 bytes
```



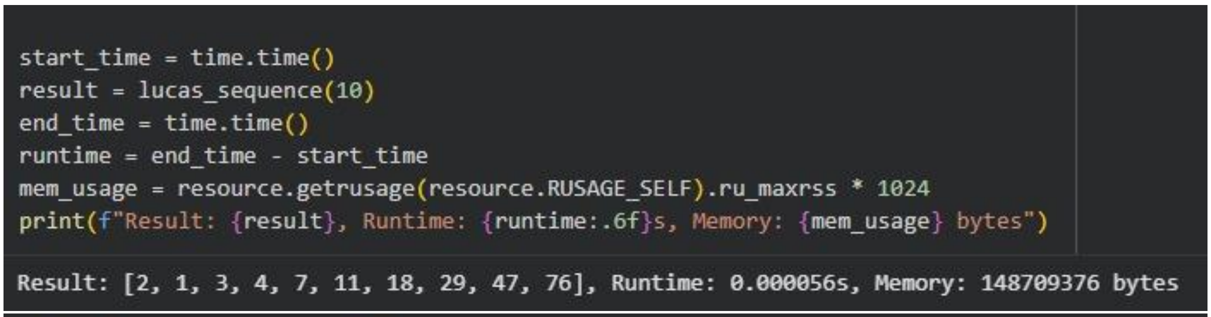
```
start_time = time.time()
result = order_mod(2, 7)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 3, Runtime: 0.000056s, Memory: 148709376 bytes
```



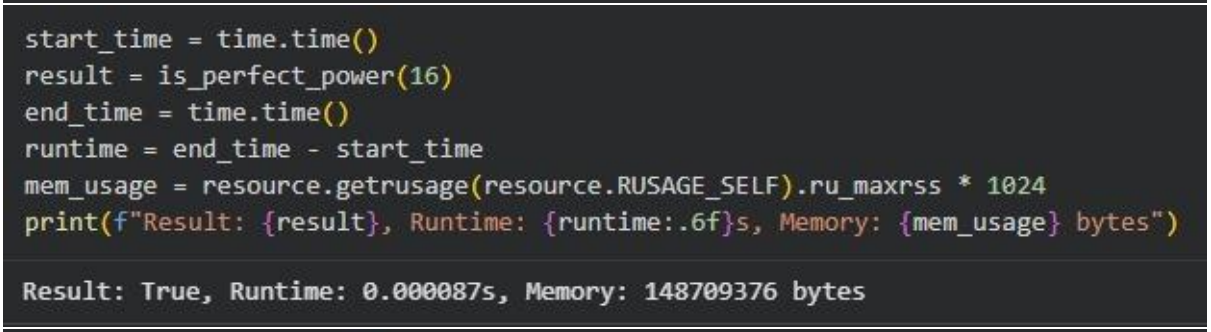
```
start_time = time.time()
result = is_fibonacci_prime(13)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: True, Runtime: 0.000065s, Memory: 148709376 bytes
```



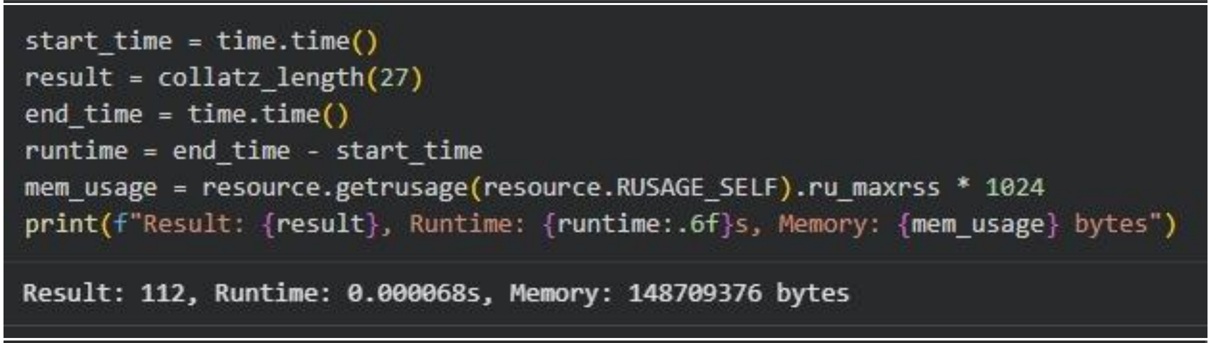
```
start_time = time.time()
result = lucas_sequence(10)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: [2, 1, 3, 4, 7, 11, 18, 29, 47, 76], Runtime: 0.000056s, Memory: 148709376 bytes
```



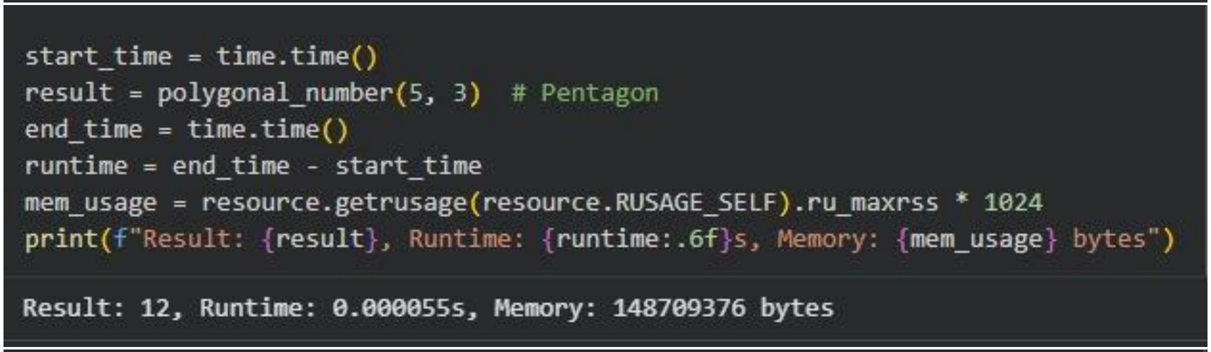
```
start_time = time.time()
result = is_perfect_power(16)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: True, Runtime: 0.000087s, Memory: 148709376 bytes
```



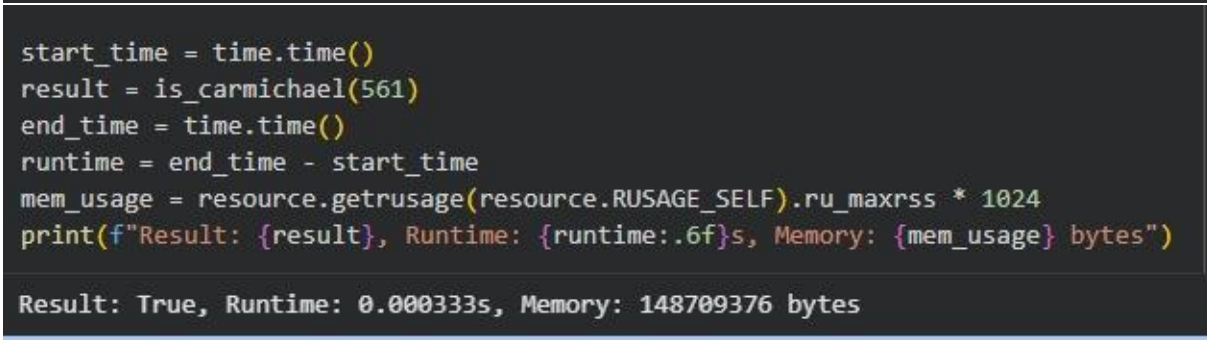
```
start_time = time.time()
result = collatz_length(27)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 112, Runtime: 0.000068s, Memory: 148709376 bytes
```



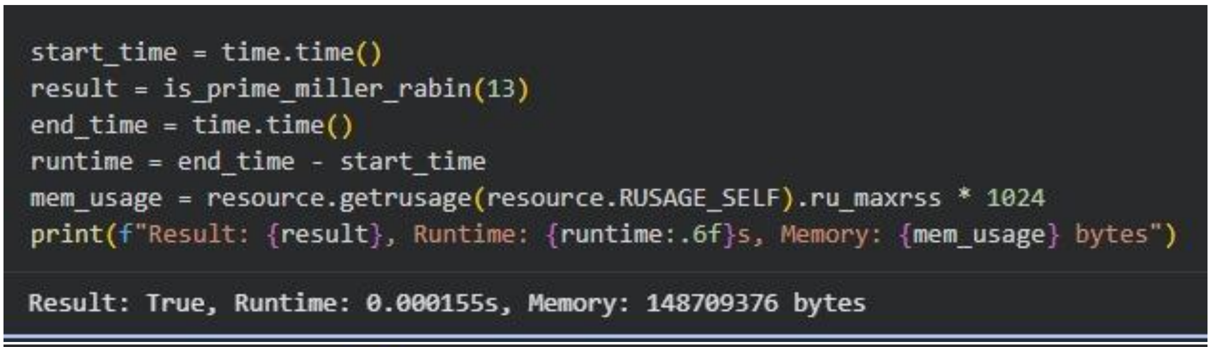
```
start_time = time.time()
result = polygonal_number(5, 3)  # Pentagon
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 12, Runtime: 0.000055s, Memory: 148709376 bytes
```



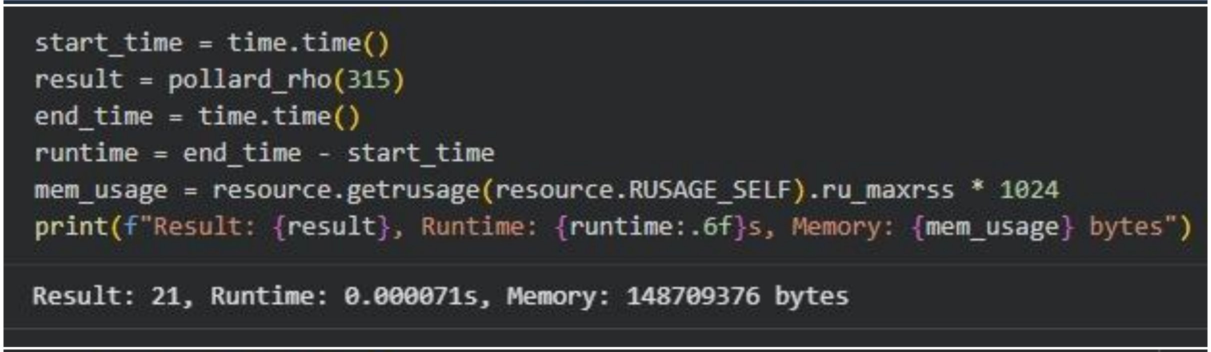
```
start_time = time.time()
result = is_carmichael(561)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: True, Runtime: 0.000333s, Memory: 148709376 bytes
```



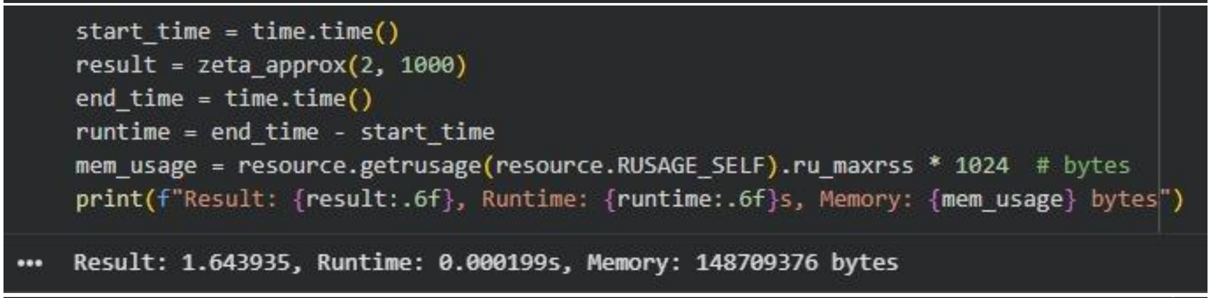
```
start_time = time.time()
result = is_prime_miller_rabin(13)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: True, Runtime: 0.000155s, Memory: 148709376 bytes
```



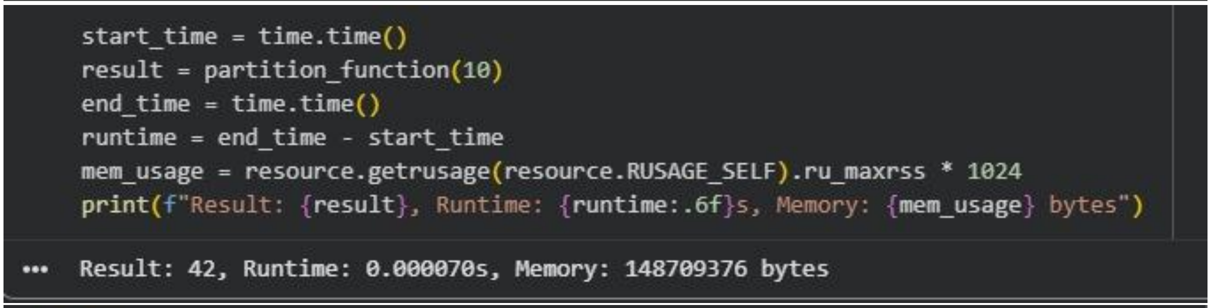
```
start_time = time.time()
result = pollard_rho(315)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 21, Runtime: 0.000071s, Memory: 148709376 bytes
```



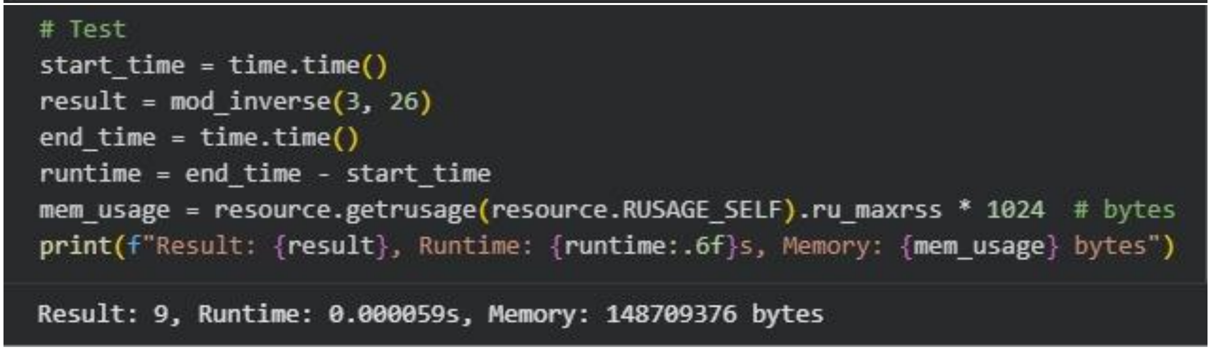
```
start_time = time.time()
result = zeta_approx(2, 1000)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024  # bytes
print(f"Result: {result:.6f}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
••• Result: 1.643935, Runtime: 0.000199s, Memory: 148709376 bytes
```



```
start_time = time.time()
result = partition_function(10)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
••• Result: 42, Runtime: 0.000070s, Memory: 148709376 bytes
```



```
# Test
start_time = time.time()
result = mod_inverse(3, 26)
end_time = time.time()
runtime = end_time - start_time
mem_usage = resource.getrusage(resource.RUSAGE_SELF).ru_maxrss * 1024  # bytes
print(f"Result: {result}, Runtime: {runtime:.6f}s, Memory: {mem_usage} bytes")
```

```
Result: 9, Runtime: 0.000059s, Memory: 148709376 bytes
```