DSC 20
Discussion Section 7

ARDA CANKAT BATI

Today's Plan

1. Talking about HW07
2. Topics Reviews:
   - Mutation / References
   - Recursion
   - Complexity
   - Higher Order functions

HW07 Questions

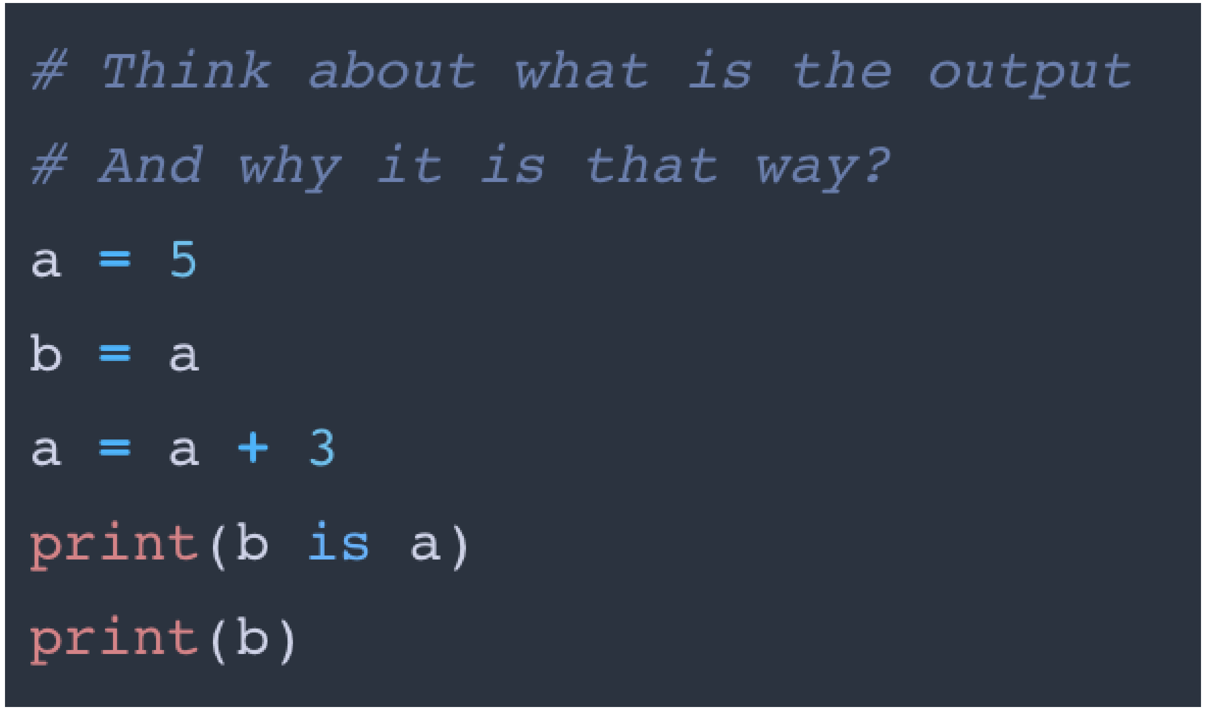
1. Let's talk about the:

   - Mutation / copying question
   - OOP question
   - Recursion question

Topic Reviews

1. Mutability / References
2. Recursion
3. Complexity
4. HOF
5. Advanced argument passing

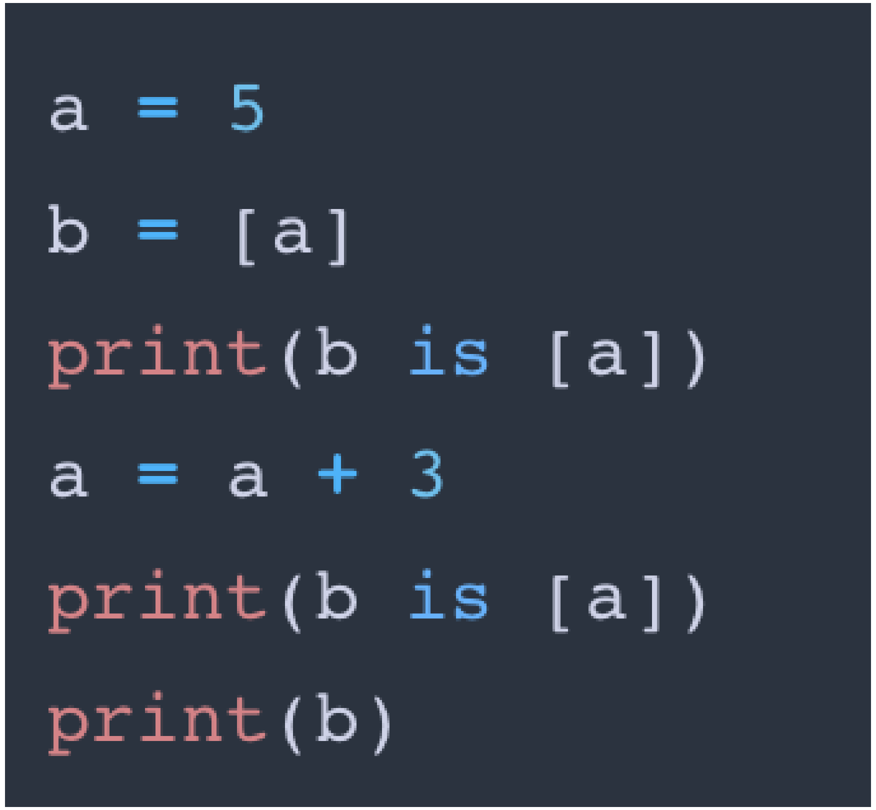
Mutability / References



```
# Think about what is the output
# And why it is that way?
a = 5
b = a
a = a + 3
print(b is a)
print(b)
```

A) True 8
B) False 8
C) True 5
D False 5

Mutability / References



```
a = 5
b = [a]
print(b is [a])
a = a + 3
print(b is [a])
print(b)
```

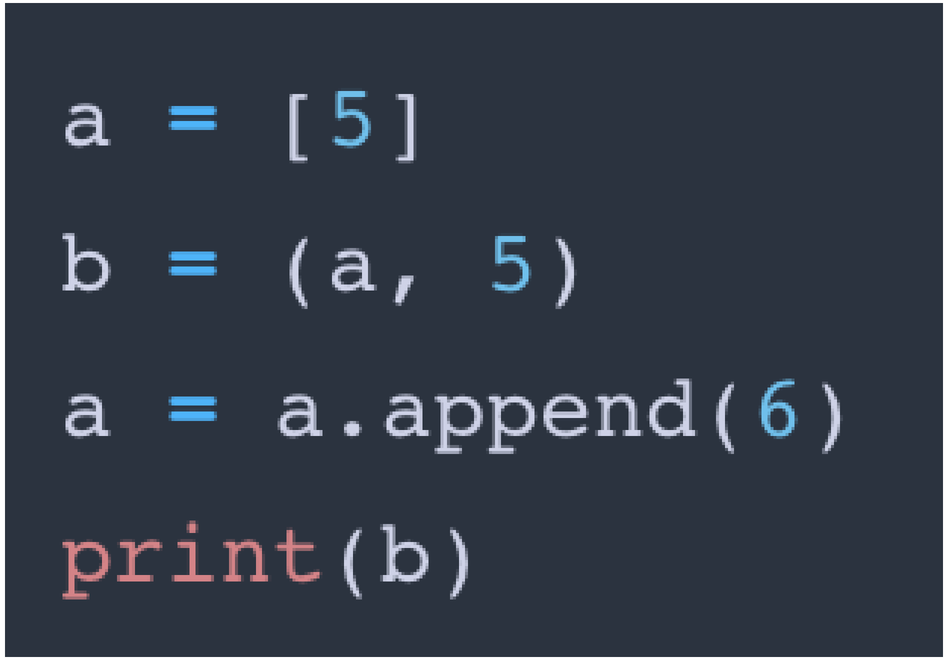
A) True, True [8]

B) True False [8]

C) False False [5]

D True False [5]

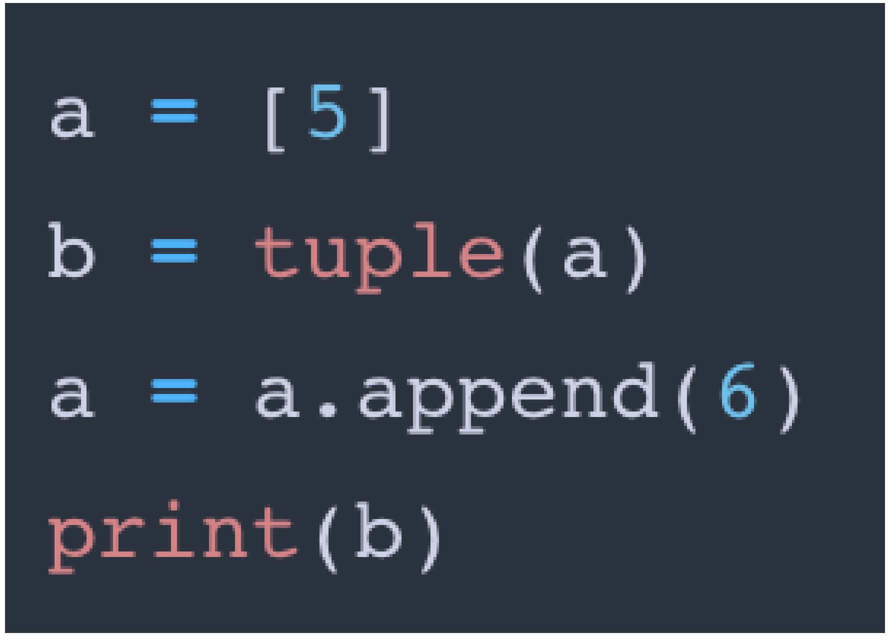
Mutability / References



```
a = [5]
b = (a, 5)
a = a.append(6)
print(b)
```

A) ([5], 5)
B) ([5,5,6])
C) ([5,6], 5)
D) Error can't hash list
E) Error, can't mutate tuple

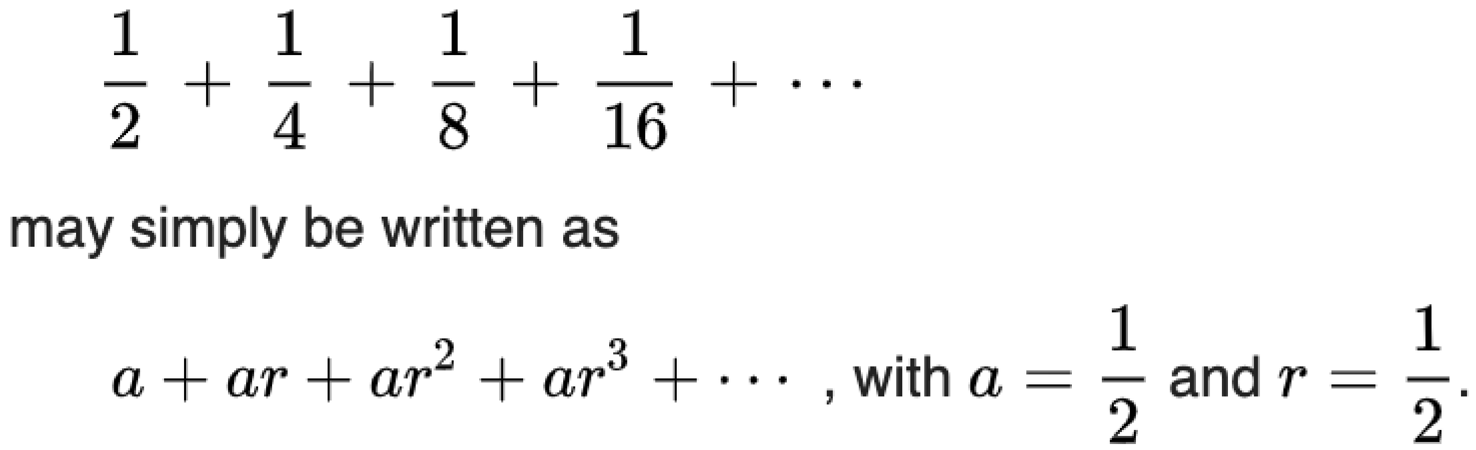
Mutability / References



```
a = [5]
b = tuple(a)
a = a.append(6)
print(b)
```

A) (5,)
B) ([5,6])
C) ([5,6], 5)
D) Error can't hash list
E) Error, can't mutate tuple

Recursion examples

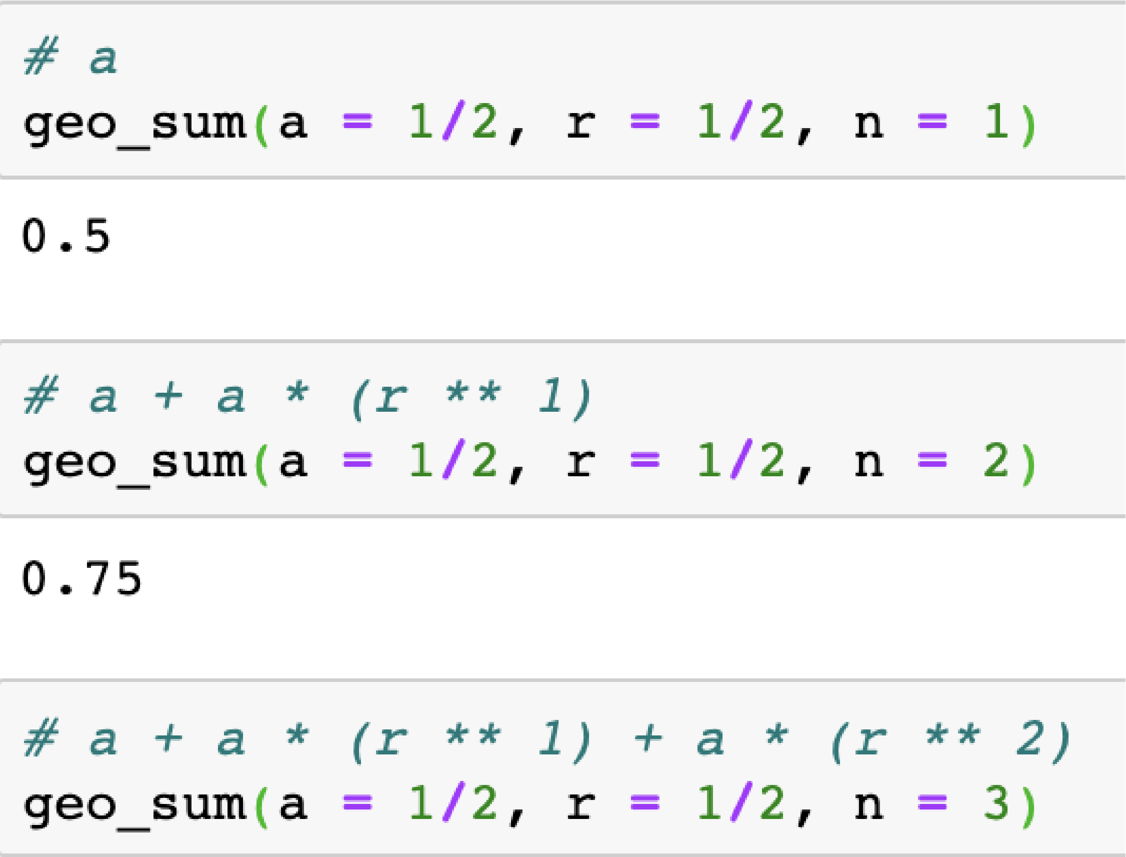


$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

may simply be written as

$$a + ar + ar^2 + ar^3 + \cdots$$, with $a = \frac{1}{2}$ and $r = \frac{1}{2}$.

Recursion examples



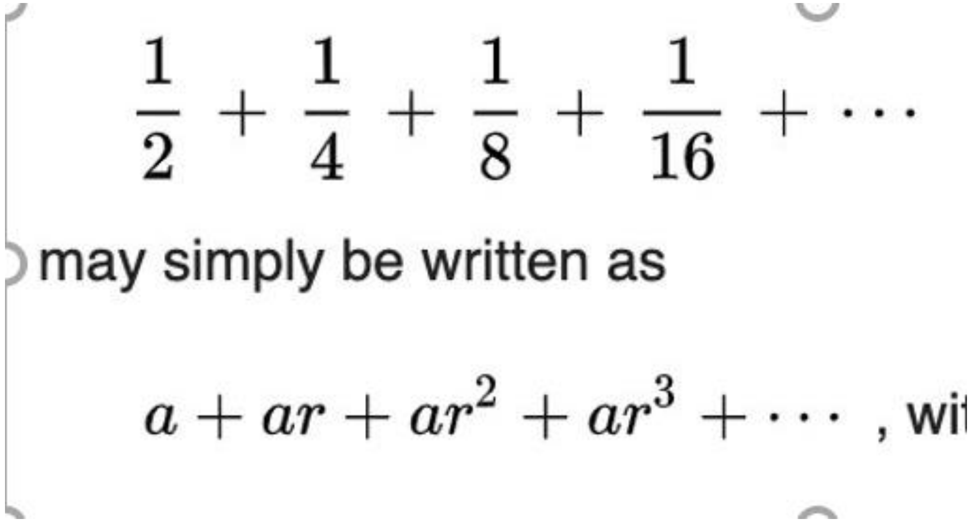
```
# a
geo_sum(a = 1/2, r = 1/2, n = 1)
```

```
0.5
```

```
# a + a * (r ** 1)
geo_sum(a = 1/2, r = 1/2, n = 2)
```

```
0.75
```

```
# a + a * (r ** 1) + a * (r ** 2)
geo_sum(a = 1/2, r = 1/2, n = 3)
```

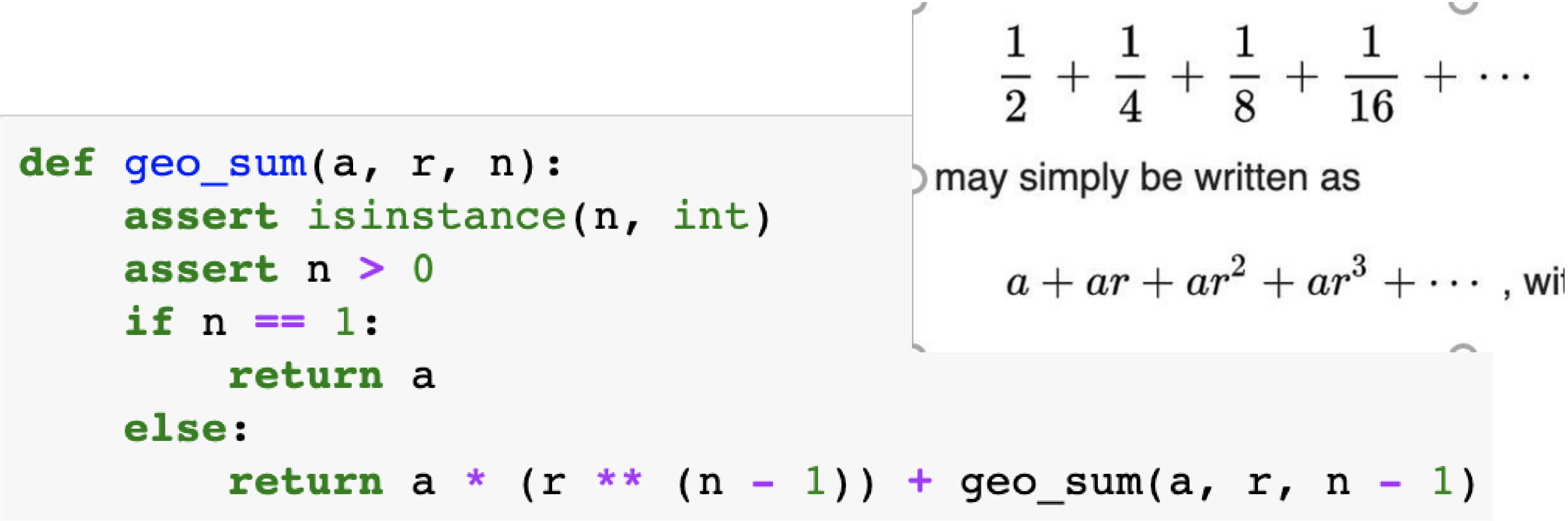


$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

may simply be written as

$$a + ar + ar^2 + ar^3 + \cdots$$, wi

Recursion examples

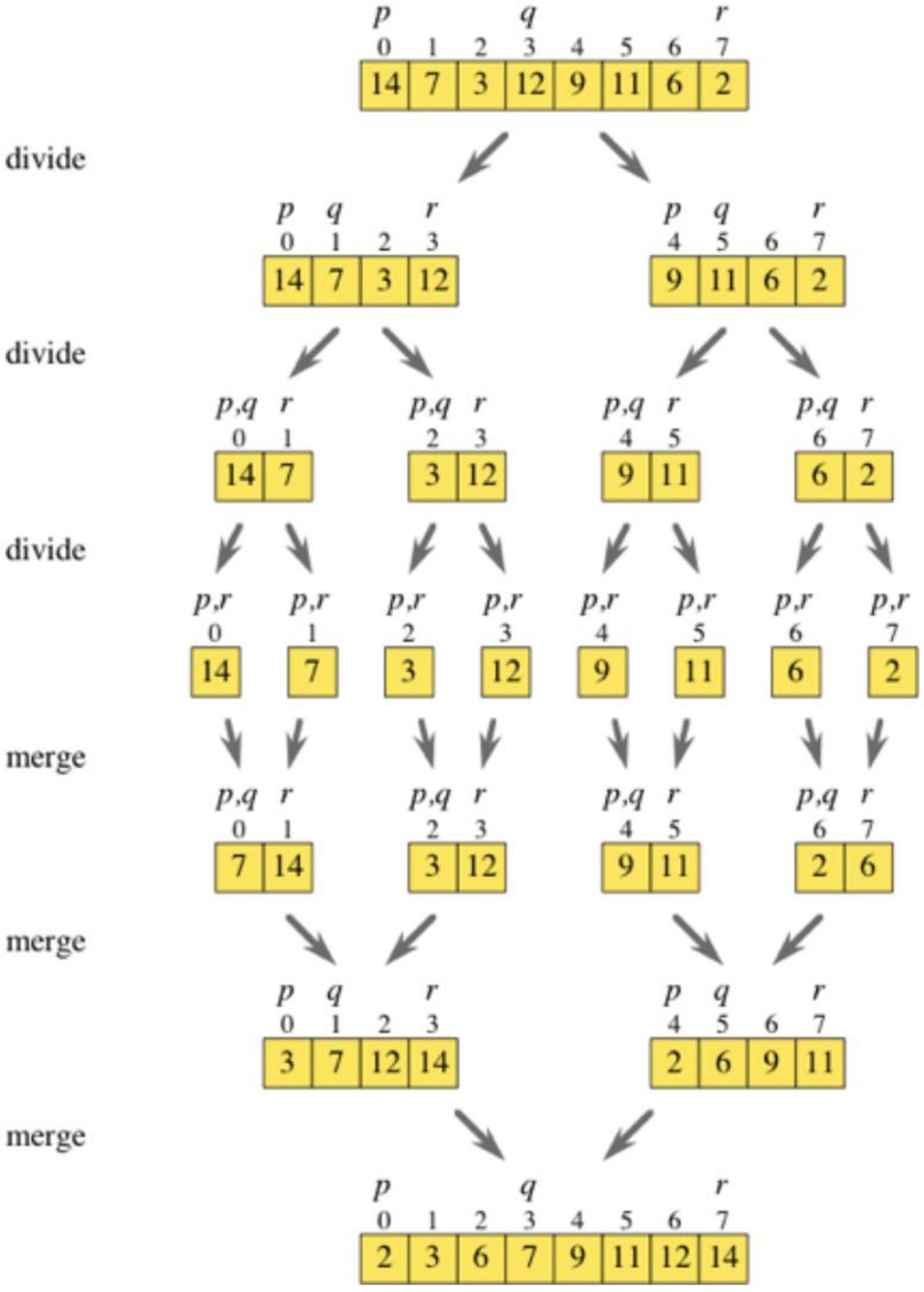


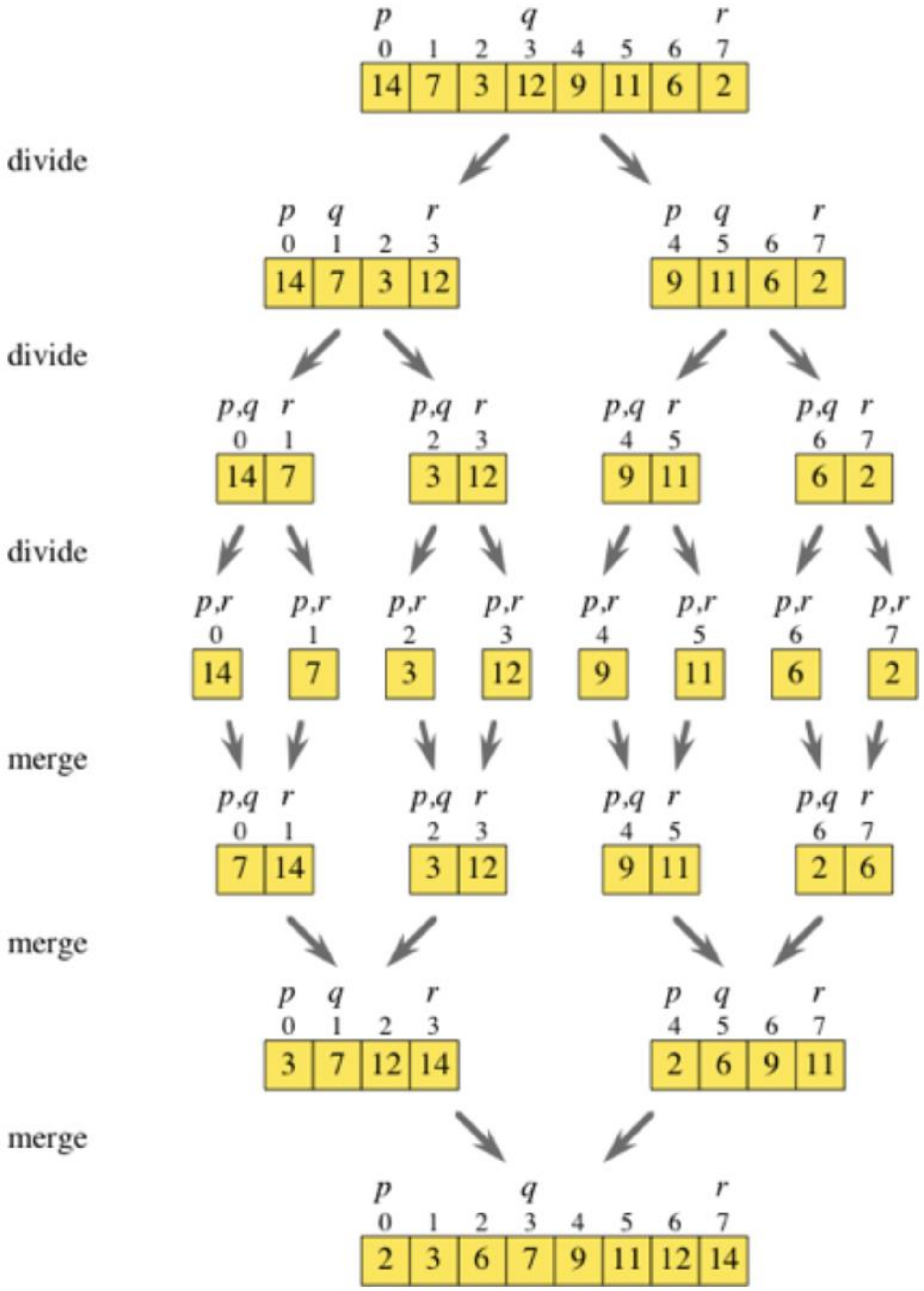
$$\frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} + \cdots$$

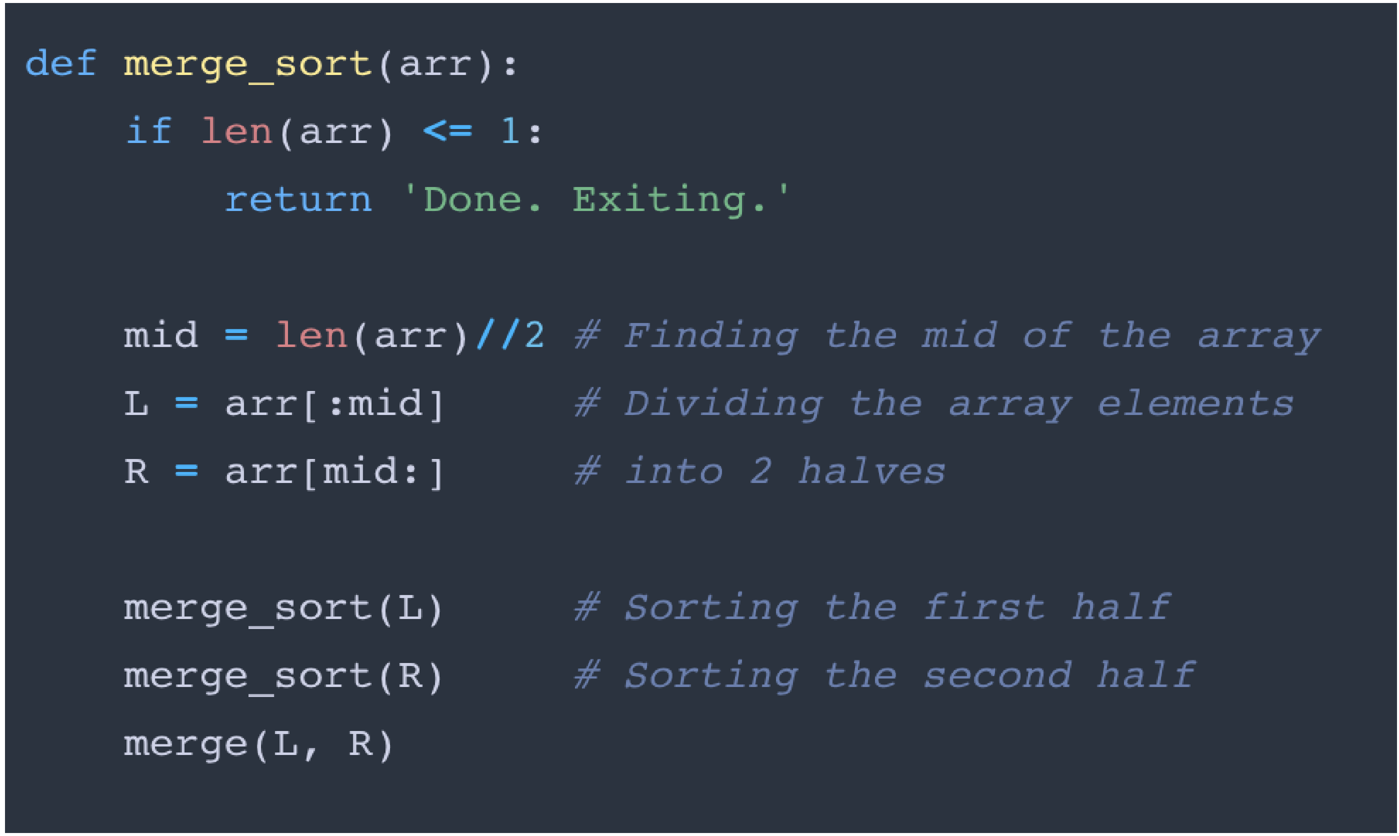
may simply be written as

$a + ar + ar^2 + ar^3 + \cdots$, wi

```
def geo_sum(a, r, n):
    assert isinstance(n, int)
    assert n > 0
    if n == 1:
        return a
    else:
        return a * (r ** (n - 1)) + geo_sum(a, r, n - 1)
```


p q r
0 1 2 3 4 5 6 7
14 7 3 12 9 11 6 2
divide
p q r
0 1 2 3
14 7 3 12
p q r
4 5 6 7
9 11 6 2
divide
p,q r
0 1
14 7
p,q r
2 3
3 12
p,q r
4 5
9 11
p,q r
6 7
6 2
divide
p,r
0
14
p,r
1
7
p,r
2
3
p,r
3
12
p,r
4
9
p,r
5
11
p,r
6
6
p,r
7
2
merge
p,q r
0 1
7 14
p,q r
2 3
3 12
p,q r
4 5
9 11
p,q r
6 7
2 6
merge
p q r
0 1 2 3
3 7 12 14
p q r
4 5 6 7
2 6 9 11
merge
p q r
0 1 2 3 4 5 6 7
2 3 6 7 9 11 12 14



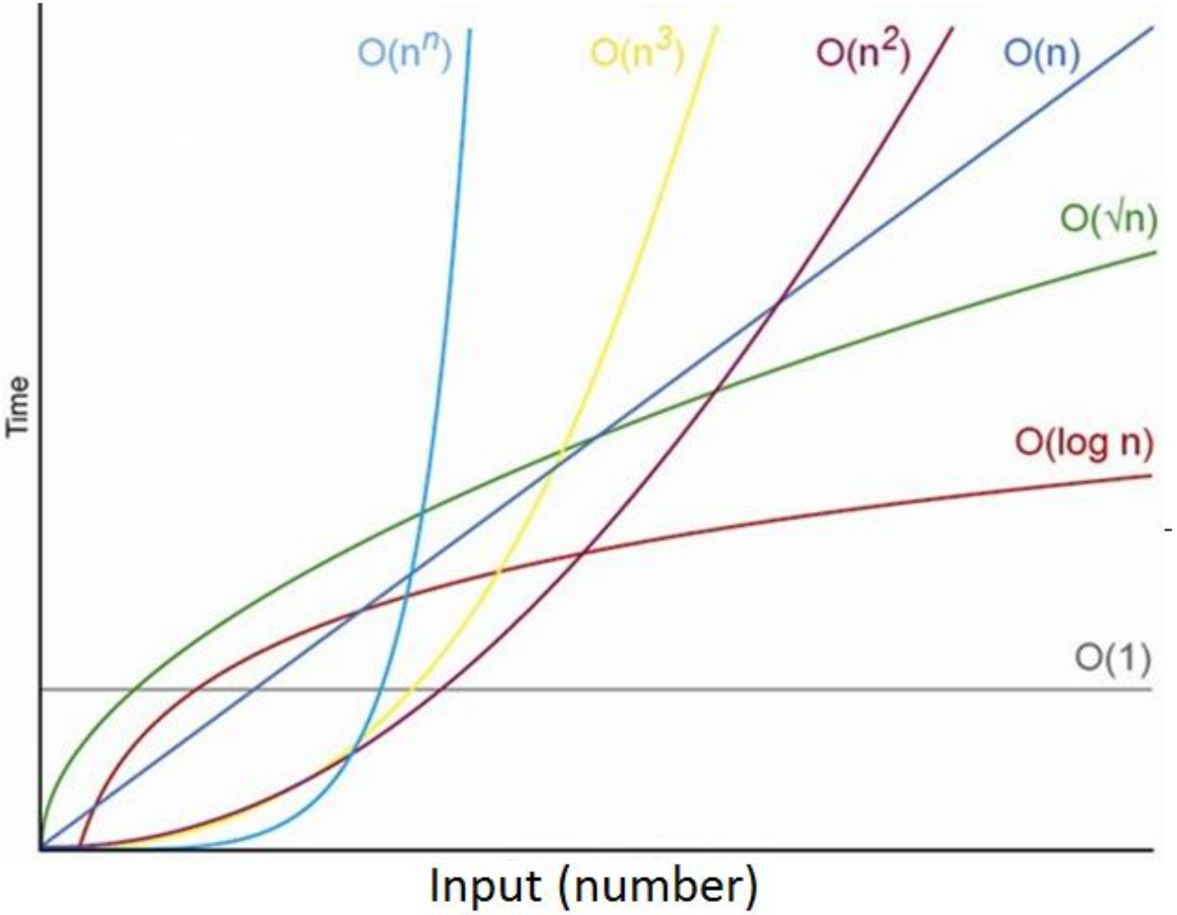


```python
def merge_sort(arr):
    if len(arr) <= 1:
        return 'Done. Exiting.'

    mid = len(arr)//2 # Finding the mid of the array
    L = arr[:mid]     # Dividing the array elements
    R = arr[mid:]     # into 2 halves

    merge_sort(L)     # Sorting the first half
    merge_sort(R)     # Sorting the second half
    merge(L, R)
```

Time Complexity



Notice how the difference between different orders are not very clear for low n.

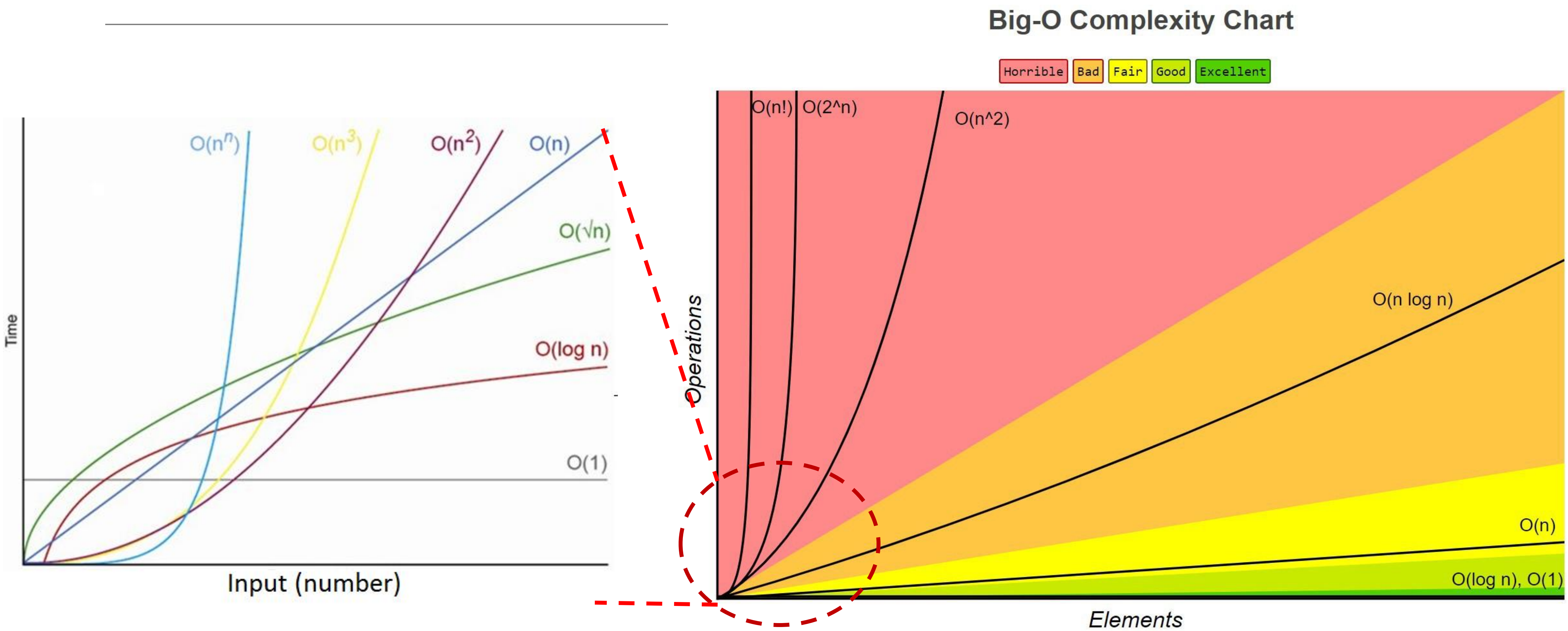
As n gets higher we start seeing the real difference between each order.

Algorithms with exponentials n^n, a^n are almost never feasible to use.

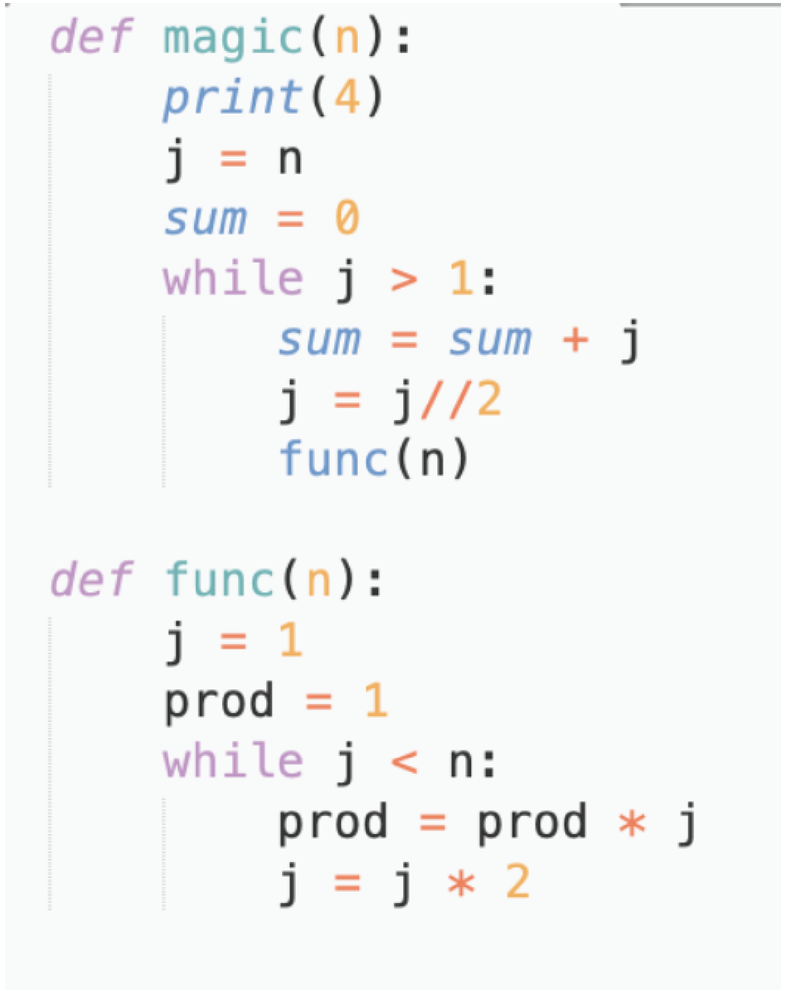
O(1) is the best, but it needs very specific structures and problem types to have such algorithms.

Usually something with log(n) will be more commonplace.

Time Complexity

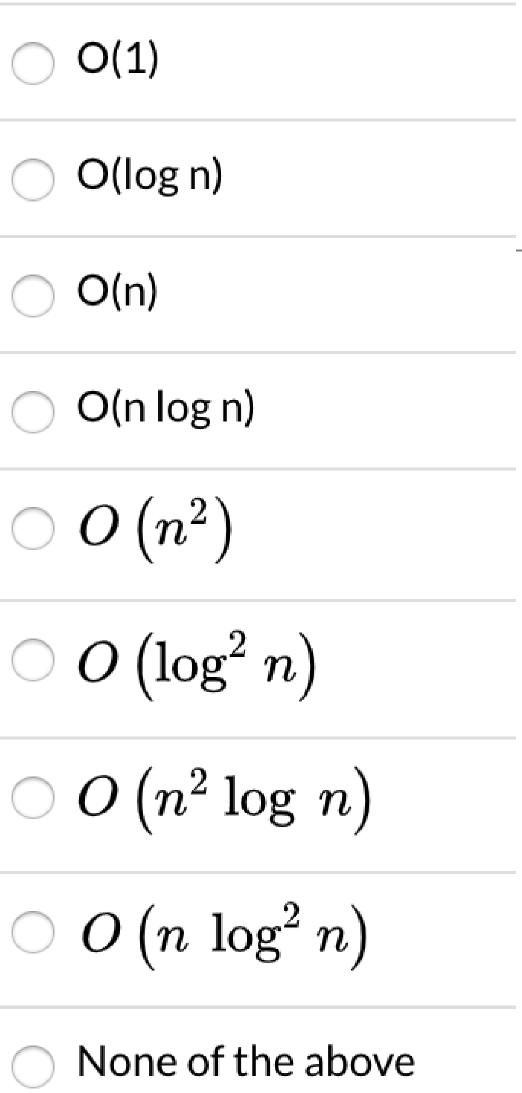


what is the complexity of the given code?



```
def magic(n):
    print(4)
    j = n
    sum = 0
    while j > 1:
        sum = sum + j
        j = j//2
        func(n)

def func(n):
    j = 1
    prod = 1
    while j < n:
        prod = prod * j
        j = j * 2
```



○ O(1)

○ O(log n)

○ O(n)

○ O(n log n)

○ $O\left(n^2\right)$

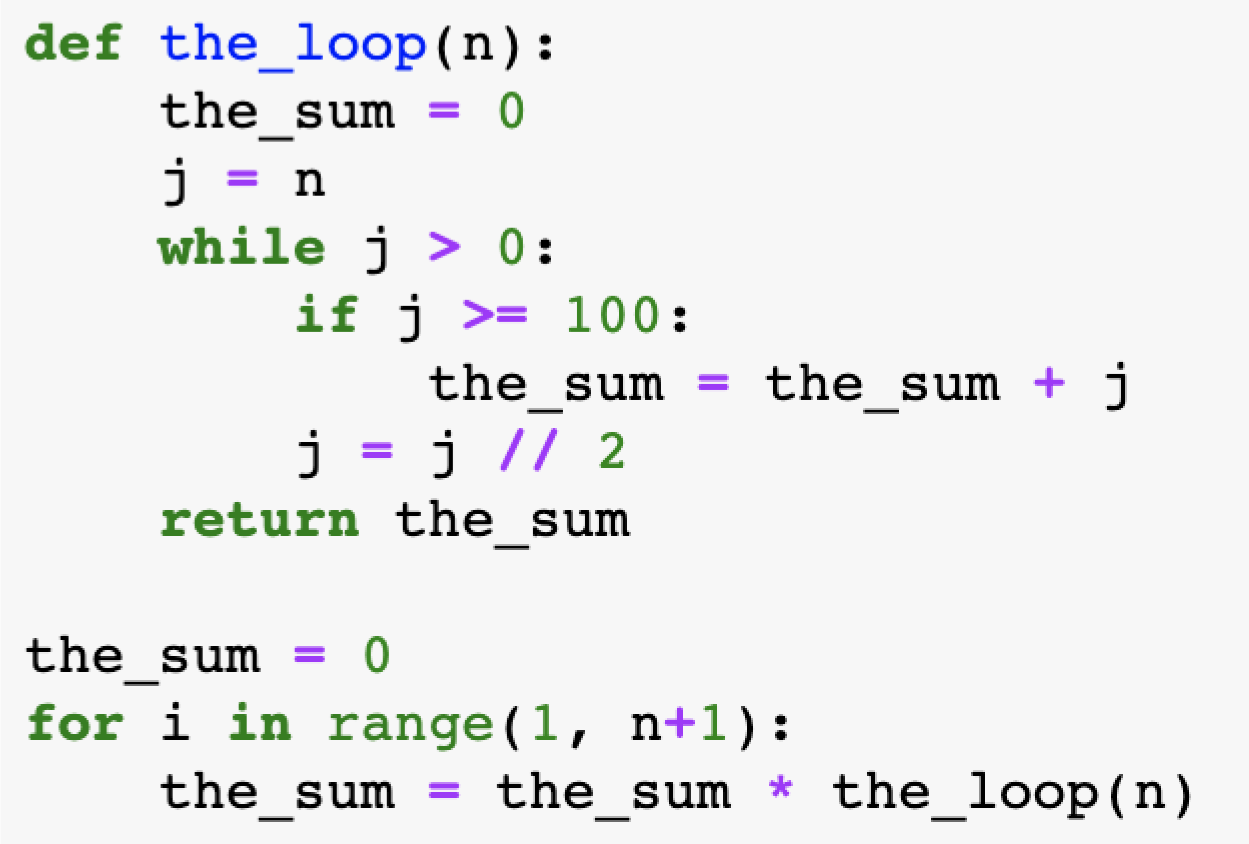
○ $O\left(\log^2 n\right)$

○ $O\left(n^2 \log\ n\right)$

○ $O\left(n\ \log^2 n\right)$

○ None of the above

Time Complexity Example 6



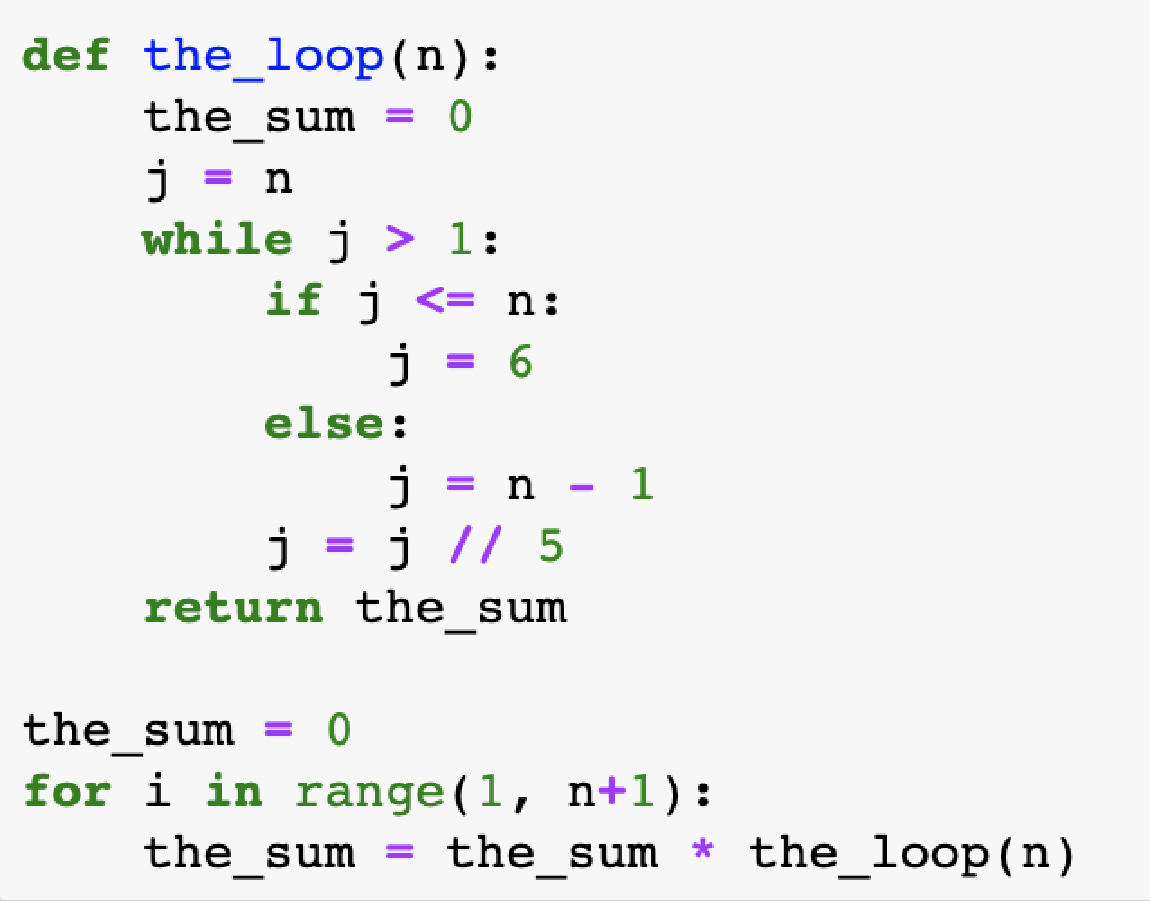
```
def the_loop(n):
    the_sum = 0
    j = n
    while j > 0:
        if j >= 100:
            the_sum = the_sum + j
        j = j // 2
    return the_sum

the_sum = 0
for i in range(1, n+1):
    the_sum = the_sum * the_loop(n)
```

iClicker Question

A. O (1)
B. O (n)
C. O (log n)
D. O (n ^ 2)
E. O (n log n)

Time Complexity Example 7



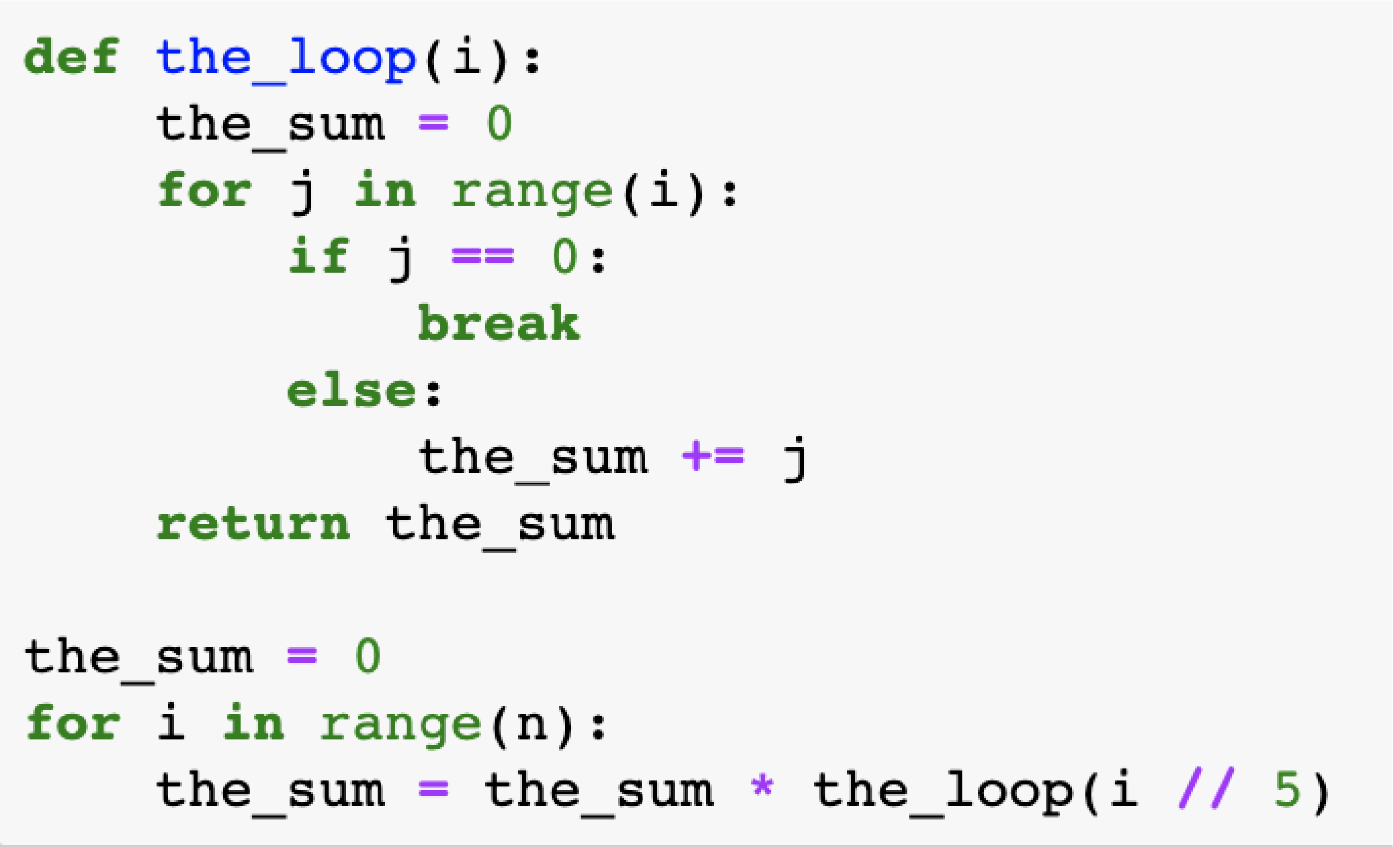
```
def the_loop(n):
    the_sum = 0
    j = n
    while j > 1:
        if j <= n:
            j = 6
        else:
            j = n - 1
        j = j // 5
    return the_sum

the_sum = 0
for i in range(1, n+1):
    the_sum = the_sum * the_loop(n)
```

iClicker Question

A. O (1)
B. O (n)
C. O (log n)
D. O (n ^ 2)
E. O (n log n)

Time Complexity Example 8



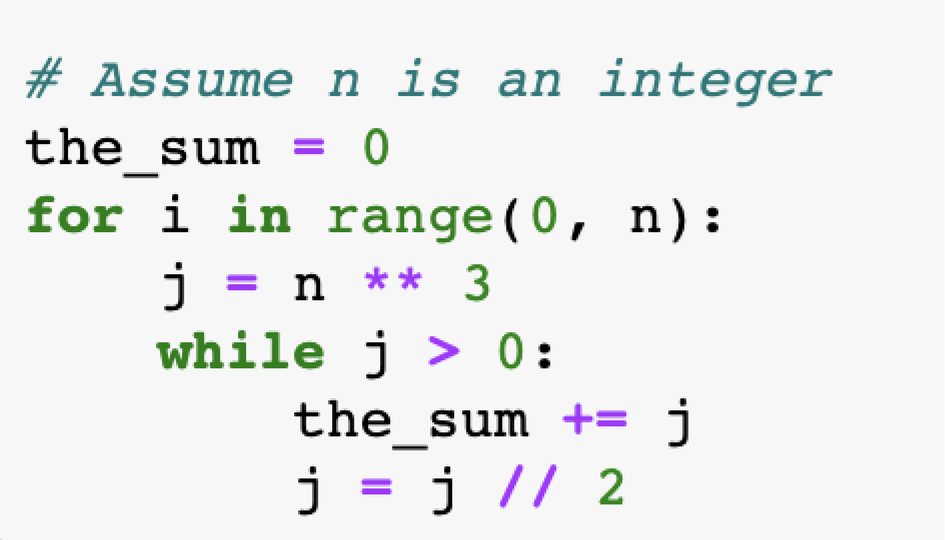
```
def the_loop(i):
    the_sum = 0
    for j in range(i):
        if j == 0:
            break
        else:
            the_sum += j
    return the_sum

the_sum = 0
for i in range(n):
    the_sum = the_sum * the_loop(i // 5)
```

iClicker Question

A. O (1)
B. O (n)
C. O (log n)
D. O (n ^ 2)
E. O (n log n)

Time Complexity Example 9



```
# Assume n is an integer
the_sum = 0
for i in range(0, n):
    j = n ** 3
    while j > 0:
        the_sum += j
        j = j // 2
```

iClicker Question

A. O (n)

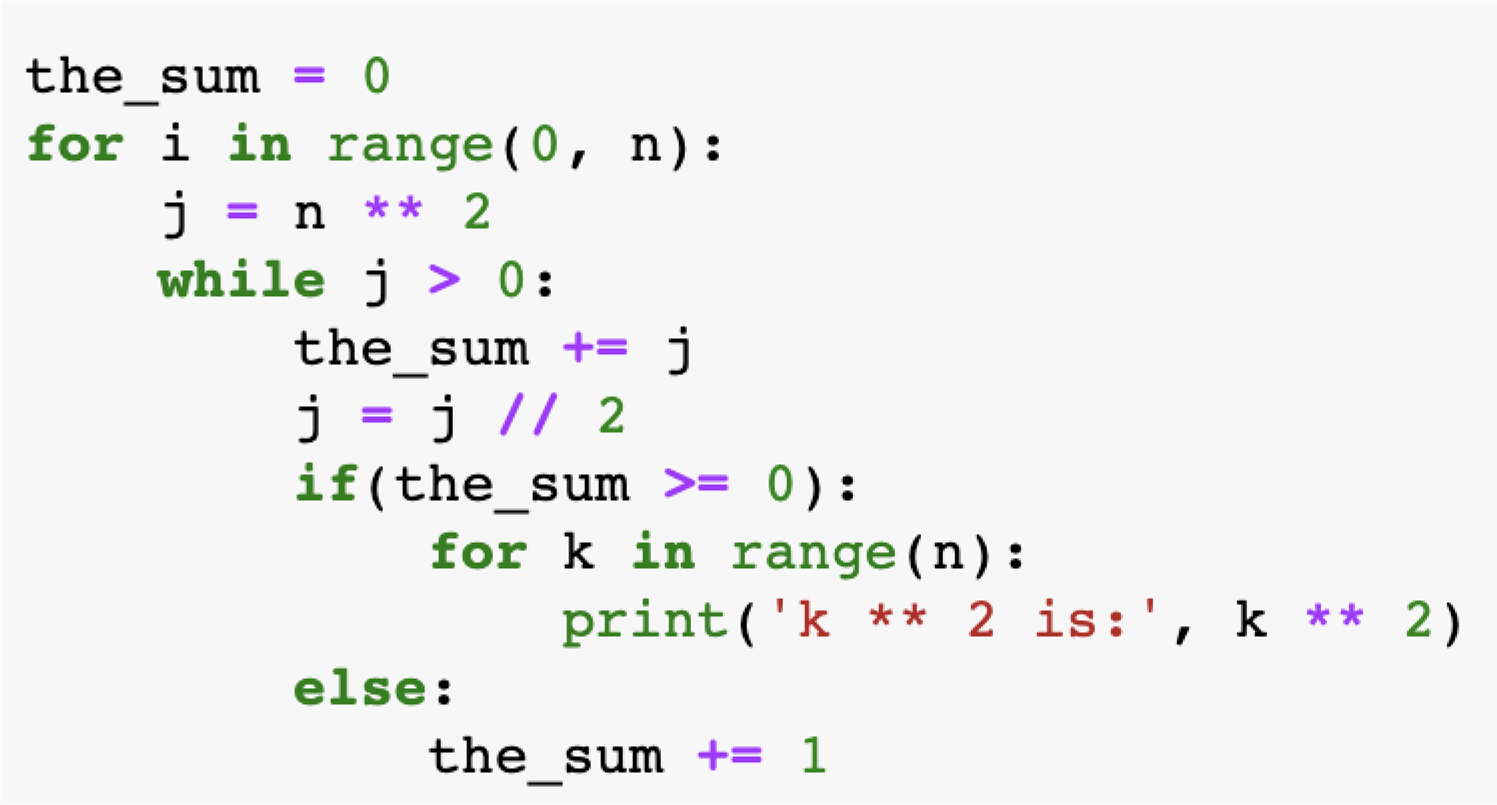
B. O (log n)

C. O (log n^3)

D. O (n log n)

E. O (n log n^3)

Time Complexity Example 10



```
the_sum = 0
for i in range(0, n):
    j = n ** 2
    while j > 0:
        the_sum += j
        j = j // 2
        if(the_sum >= 0):
            for k in range(n):
                print('k ** 2 is:', k ** 2)
        else:
            the_sum += 1
```

iClicker Question

A. O (n)

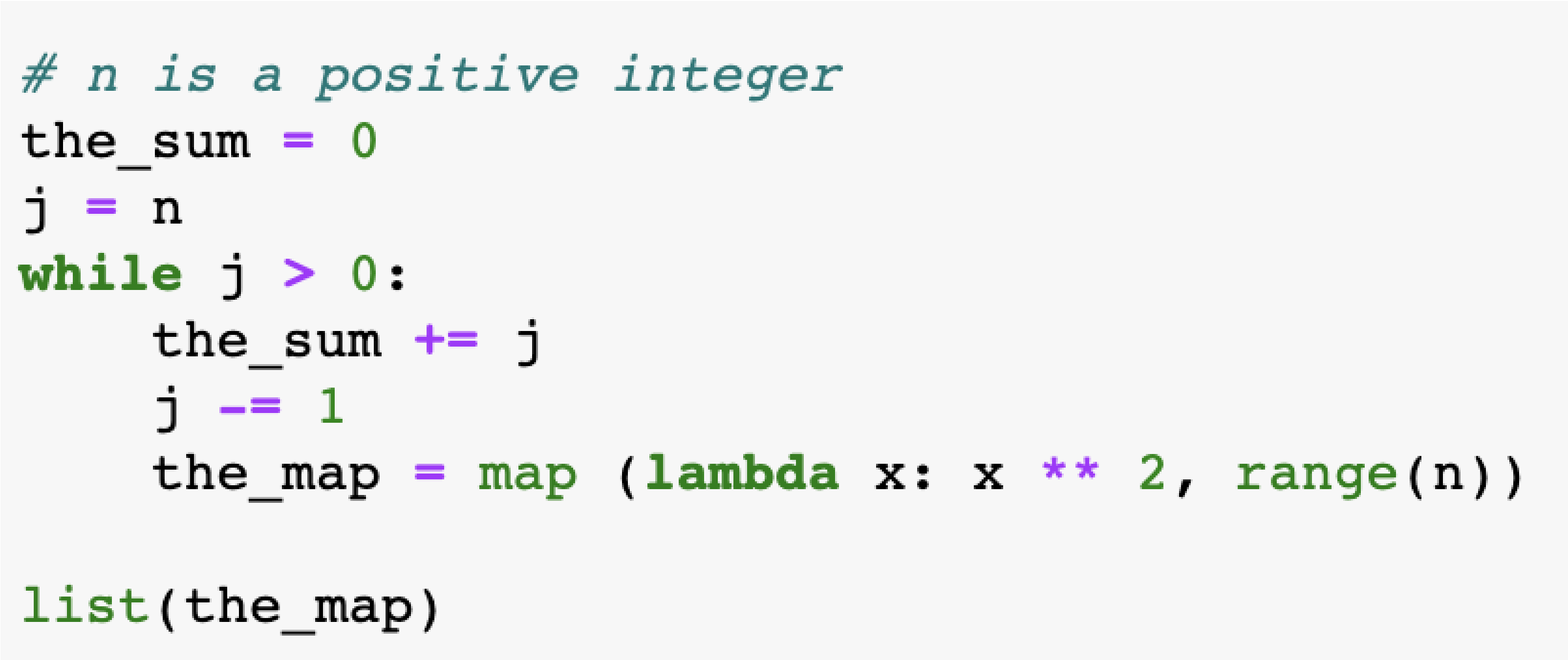
B. O (n ^ 2)

C. O (log n)

D. O (n^2 log n)

E. O (n log n)

Time Complexity Example 11



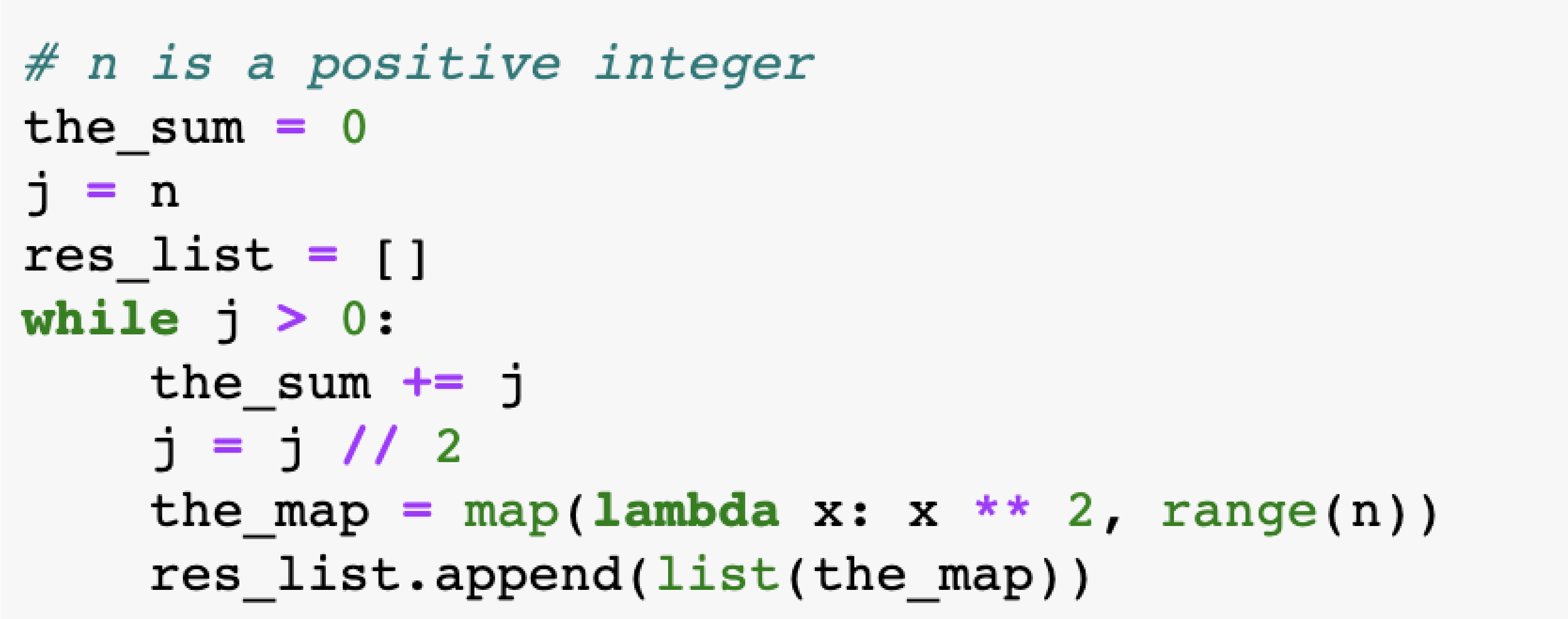
```
# n is a positive integer
the_sum = 0
j = n
while j > 0:
    the_sum += j
    j -= 1
    the_map = map (lambda x: x ** 2, range(n))

list(the_map)
```

iClicker Question

A. O (n)
B. O (n ^ 2)
C. O (log n^2)
D. O (n log n)
E. Something Else

Time Complexity Example 12

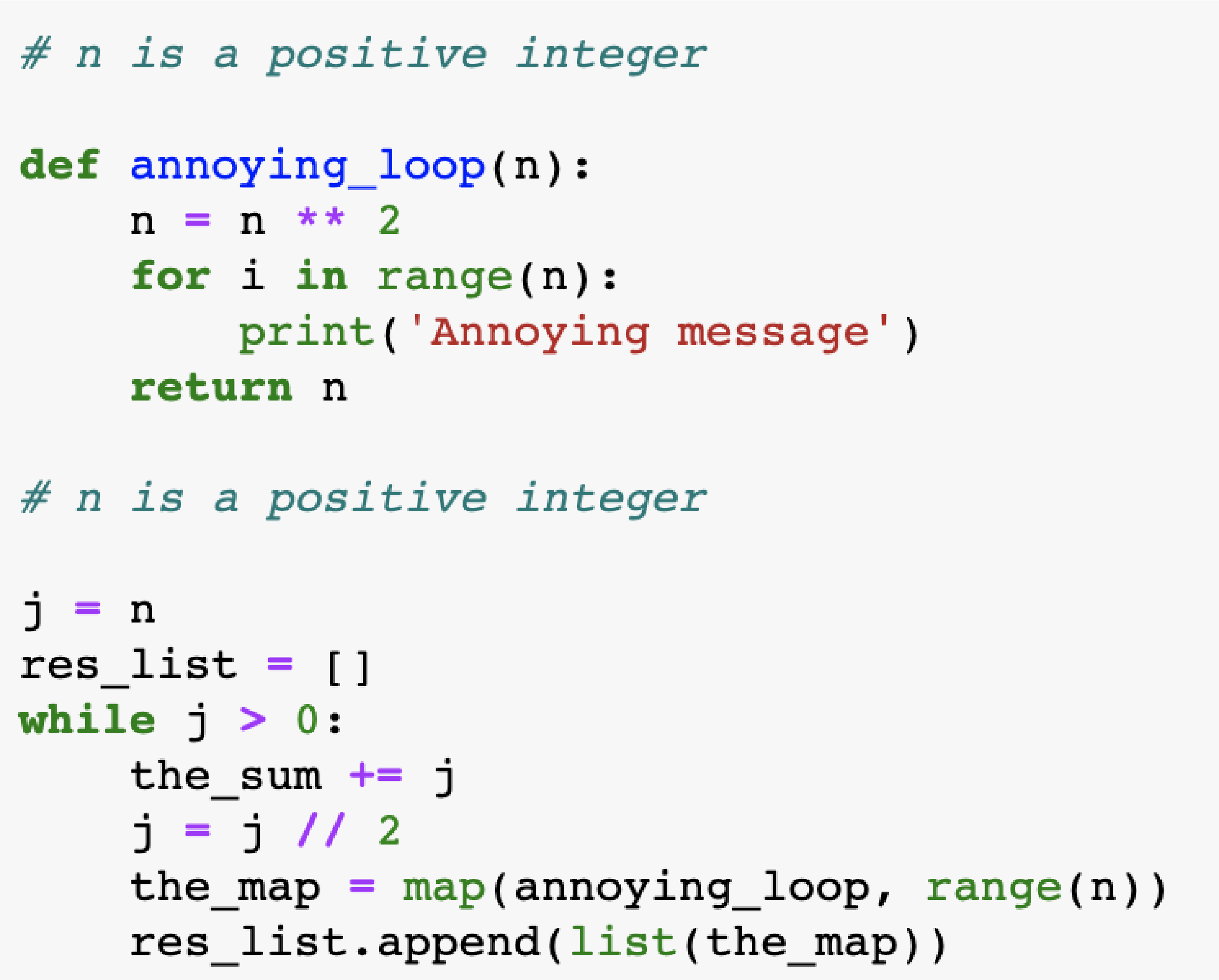


```
# n is a positive integer
the_sum = 0
j = n
res_list = []
while j > 0:
    the_sum += j
    j = j // 2
    the_map = map(lambda x: x ** 2, range(n))
    res_list.append(list(the_map))
```

iClicker Question

A. O (n)
B. O (n ^ 2)
C. O (log n)
D. O (n log n)
E. O (n^2 log n)

Time Complexity Example 13



```
# n is a positive integer

def annoying_loop(n):
    n = n ** 2
    for i in range(n):
        print('Annoying message')
    return n

# n is a positive integer

j = n
res_list = []
while j > 0:
    the_sum += j
    j = j // 2
    the_map = map(annoying_loop, range(n))
    res_list.append(list(the_map))
```

iClicker Question

A. O (n)
B. O (n ^ 2)
C. O (log n)
D. O (n log n)
E. O (n^2 log n)