

Recursion vs Iteration

Textbook

Recursion vs Iteration



As you learned in the last lesson's story of the recursive Santa Claus, recursion can make complex tasks much easier. If Santa were to use an iterative process to deliver all the presents on Christmas, it would be a long, hard day for him!

Using his elves and a recursive algorithm, Santa was able to assign jobs to his elves and get all the presents delivered quickly and easily on Christmas.

Iterative Factorial

One of the most common ways to learn about recursion is to write a factorial program. As a reminder, getting the factorial of a number involves multiplying all the numbers from 1 to the number. For example, the factorial of 4 is $1 * 2 * 3 * 4$, which equals 24.

One way to get the factorial of a number is using an iterative process. The code for this method looks like this:

```
1 def iterative_factorial(n):
2     result = 1
3     for i in range(1, n + 1):
4         result *= i
5     return result
6
```

```
7 print(iterative_factorial(4))
```

Try it!

This code gets an integer input from the user, then uses a for loop to multiply all the numbers for the factorial. This is a good method to use, but recursion is much faster.

Recursive Factorial

A recursive factorial function is shown below.

```
1 def factorial(n):
2     if n <= 1:
3         return 1
4     else:
5         return n * (factorial(n - 1))
6
7 print(factorial(4))
```

Try it!

Notice how the "else" branch calls the function it's already in? This is recursion. It will keep calling the function on $n-1$ until $n == 1$, then it will multiply from 1 through the number of the factorial.

- Try Googling the word "recursion." They have a funny Easter egg there, can you find what it is?

Checkpoint

Recursion vs. Iteration

Write your own recursive factorial function. It should take in one integer as input, and use recursion to print its factorial.

For example:

Input: 5

Output: 120

Another example:

Input: 10

Output: 3628800

Questions (3)

1. Which method of programming is faster for the computer to calculate?

MULTIPLE CHOICE

Choose the correct answer:

- A. iteration
- B. recursion

2. True or False: There are usually many ways to get to the same answer in programming. Some are faster or more efficient for the computer to calculate than others.

MULTIPLE CHOICE

Choose the correct answer:

- A. True
- B. False

3. What approach is the following code an example of?

MULTIPLE CHOICE

```
def answer(n): if n <= 1: return 1 else: return n + (answer(n - 1)) print(answer(10))
```

Choose the correct answer:

- A. Iteration
- B. Recursion

Challenges (2)

1. Add the List

Write a recursive function that takes in a list as input and adds all the numbers in the list. Print the sum of all the numbers.

For example:

Inputs: 2 2 2 2

Output: 8

Another example:

Inputs: 2 4 6 8

Output: 20

Hint: Here is how to create a list from an input:

```
list_of_nums = [int(n) for n in input().split()]
```

2. Fibonacci

In the Fibonacci sequence, the next number is found by adding up the two numbers before it.

Here is an example:

0 1 1 2 3 5 8 13 21 34

This sequence is often referred to as The Golden Ratio. This ratio can often be found in nature in examples such as the length of the segments of your fingers in relation to each other and in relation to your hand, up to your elbow, etc.

1. Write a recursive function that takes in one integer as input and finds that many numbers in the Fibonacci sequence.
2. Create an empty list and append each item into the list.
3. Print the completed list of the Fibonacci Sequence.

For example:

Input: `5`

Output: `[0, 1, 1, 2, 3]`

Another example:

Input: `10`

Output: `[0, 1, 1, 2, 3, 5, 8, 13, 21, 34]`

Note: The recursive function will need to be called from within a `for i in range()` function.

Answer Keys & Solutions

Checkpoint Solutions

Recursion vs. Iteration

```
1 def factorial(num):
2     if (num == 1 or num == 0):
3         return 1
4     else:
5         return (num * factorial(num - 1))
6
7 num = int(input())
8 print(factorial(num))
```

Questions

1. Which method of programming is faster for the computer to calculate?

MULTIPLE CHOICE

Correct Answer:

A. iteration

✗ Incorrect

B. recursion

✓ Correct

Explanation:

A function that calls itself is much faster than an ordinary loop.

2. True or False: There are usually many ways to get to the same answer in programming. Some are faster or more efficient for the computer to calculate than others.

MULTIPLE CHOICE

Correct Answer:

A. True

✓ Correct

B. False

✗ Incorrect

Explanation:

Coding is highly creative and often has many approaches to get a correct solution.

3. What approach is the following code an example of?

MULTIPLE CHOICE

Correct Answer:

A. Iteration

✗ Incorrect

B. Recursion

✓ Correct

Explanation:

Recursion is when a function can run many times within itself.

Challenges

1. Add the List

Solution:

```
1 list_of_nums = [int(n) for n in input().split()]
2
3 def add_list(my_list):
4     if len(my_list) == 1:
5         return my_list[0]
6     else:
7         return my_list[0] + add_list(my_list[1:])
8
9 print(add_list(list_of_nums))
```

2. Fibonacci

Solution:

```
1 final = []
2
3 def fibonacci(num):
4     if num <= 1:
5         return num
6     else:
7         return(fibonacci(num-1) + fibonacci(num-2))
8
9 length = int(input("Enter how many numbers you want to see: "))
10
11 for i in range(length):
12     final.append(fibonacci(i))
```

```
13 print(final)
```