Recursion

recurrere (L): to run back

**Strategy**: Find the answer by finding the answer to a “smaller” problem of the same type. Have the function call itself!

**Three laws of recursion:**

1. A recursive algorithm must have a **base case**. The base case has a known solution that does not require recursion.
2. A recursive algorithm must make progress toward the base case.
3. A recursive algorithm must call itself, recursively.

**Important point:** Any problem solvable by recursion can be solved by iteration (and vice versa).

**Cost of recursion:** Time and memory space to make the calls.

**Payoff**: Sometimes (but not always) recursive solutions are simpler and more elegant than iterative ones.

**Example:**

def iterative\_search(target, a\_list):

for x in a\_list:

if x == target:

return True

else:

return False

def recursive\_search(target, a\_list):

if a\_list == []:

return False

elif a\_list[0] == target:

return True

else:

return recursive\_search(a\_list[1:], target)

**Example 1**: Factorial of *n*, n!

Pseudocode

Factorial(n) =

Do a ***slice trace:*** for each recursive call, show the computed *return value* and the values of any function parameters or local variables relevant to the computation.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Main method: |  |  |  |  |
| print(factorial(5)) | |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| factorial |  |  |  |  |
|  | *return*: |  | n: |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| a | b | l | e | w | a | s | i | e | r | e | i | s | a | w | e | l | b | a |

**Example 2**: Is a string S a palindrome?

Recursive approach: S is a palindrome only if the first and last character are equal and, when the first and last character are removed, we still have a palindrome.

Pseudocode

palindrome(s) =

Complete the slice trace below:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Main method: |  |  |  |  |
| print(palindrome(‘stunts’)) | |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
| palindrome |  |  |  |  |
|  | *return*: |  | s: |  |

**Example 3**: Perform a *binary search* of a list for a target item.

Pseudocode

binary\_search(target, a\_list) =

Complete the slice trace below:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Main method: |  | |  |  |  |  |  |
| x = [1, 5, 24, 32, 100]  print(binary\_search(5, x)) | | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
|  |  | |  |  |  |  |  |
| binary\_search(target, a\_list) | |  |  |  |  |  |  |
|  | *return*: | |  | target: |  | a\_list: |  |