Intro to Recursion

COSC 101: Intro to Computing I
November 15, 2017
What is recursion?
What is a base case?
What is the output of the following code?

def downup(s):
    if len(s) <= 1:
        print(s)
    else:
        print(s)
        downup(s[:-1])
        print(s)

downup('hello')
Write a **non-recursive** function to print the same pattern.

```python
>>> downup_not_recursive('hello')
hello
hell
hel
he
h
he
hel
hell
hello
```
Write a non-recursive function to print the same pattern.

```python
def downup_not_recursive(s):
    # print down version of s
    for idx in range(len(s)):
        print(s[:len(s)-idx])

    # print up version of s
    for idx in range(len(s)-1):
        print(s[:idx+2])
```

Writing Recursive Functions

1. Write the docstring first (*it really does help*)

2. Figure out the base case (often 0, 1, or "")

3. For the recursive case:
   
   i. **Divide:** Break into two pieces, a simple piece to handle now and a “harder” piece that is a smaller version of the same problem

   ii. **Recurse:** Make a recursive call with the harder piece and “have faith” that it will come together correctly

   iii. **Combine:** Put the result of the recursive call and the smaller piece together into a complete solution
Write a new recursive function to return the same pattern.

```python
>>> print(downup_return('hello'))
hello
hell
hel
he
h
he
hel
hell
hello
```
Write a new recursive function to return the same pattern.

def downup_return(s):
    # base case
    if len(s) <= 1:
        return s
    # recursive case
    else:
        # divide
        simple = s
        harder = s[:-1]
        # recurse
        recursed = downup_return( harder )
        # combine
        combined = simple + '
' + recursed + '
' + simple
    return combined
Write a recursive function to compute factorials.

A factorial of \( n \) is the product of integers from 1 to \( n \):

\[
  n! = n \times (n - 1) \times (n - 2) \times \cdots \times 2 \times 1
\]

```python
def factorial(n):
    # base cases
    if n <= 1:
        return 1

    # recursive case
    return n * factorial(abs(n)-1)
```
Write a recursive function `count_e` that takes a string `s` and returns the number of times 'e' occurs in `s`.

```python
def count_e(s):
    ''' (str) -> int
    Returns the number of times the letter e occurs in s.
    >>> count_e('madeline')
    2
    >>> count_e("here are some 'e's!")
    5
    >>> count_e('"")
    0
    '''
```
Write a recursive function `count_e` that takes a string `s` and returns the number of times 'e' occurs in `s`.

```python
def count_e(s):
    if s == '':
        # base case
        return 0
    simple = s[0]
    # divide
    harder = s[1:]
    result = count_e(harder)  # recurse
    if simple == 'e':
        # combine
        return 1 + result
    if:
        return result
```
Write a recursive function `reverse` that takes a string `s` and returns the string in reverse.

```python
def reverse(s):
    ''' (str) -> str
    Returns the string s reversed.
    >>> reverse('madeline')
    'eniledam'
    >>> reverse("here are some 'e's!"
    's'e' emos era ereh'
    >>> reverse('')
    ''
    '''
```

Write a recursive function `reverse` that takes a string `s` and returns the string in reverse.

```python
def reverse(s):
    if s == '':  # base case
        return ''
    simple = s[-1]  # divide
    harder = s[:-1]
    result = reverse(harder)  # recurse
    return simple + result  # combine
```