Lecture 7: Practice with Strings
What does \texttt{s} equal after the following operations?

```python
>>> s = "the rain in spain stays mainly on the plain"
>>> s[3]

>>> s[:3]

>>> s[4:]

>>> s[4:8]

>>> s[7:3:-1]

>>> s[:, -1]
```
Predicting operations on \texttt{s}

```python
>>> s = "the rain in spain stays mainly on the plain"
>>> s[3]
', '
>>> s[:3]
'the'
>>> s[4:]
'rain in spain stays mainly on the plain'
>>> s[4:8]
'rain'
>>> s[7:3:-1]
'niar'
>>> s[::-1]
'nialp eht no ylniam syats niaps ni niar eht'
```
split and join  Write a function `totab` that given a comma delimited string like "name,yob,age,weight" returns a tab delimited string like "name\tyob\tage\tweight".

upper and lower  Write a function called `capitalize` that given a string returns the same string but with the first character capitalized and the remaining characters in lowercase. For example, `capitalize('pURPle')` returns 'Purple'.

find  Write a function called `begins` that given a string `s` and a prefix `pre` returns True if and only if `s` begins with `pre`.

find and len  Write a function called `ends` that given a string `s` and a suffix `suf` returns True if and only if `s` ends with `suf`
def capitalize(s):
    """return a capitalized version of s"""
    return (s[0].upper + s[1:].lower())
```python
def totab(s):
    """replace the commas in s with tabs"""
    return \t.join(s.split("",""))
```
def begins(s, pre):
    """returns True if and only if s begins with pre"""
    return s.find(pre) == 0
def ends(s, suf):
    """returns True if and only if s ends with suf"""
    loc = len(s) - len(suf)
    return s.find(suf, loc) == loc
A string is called a double string when it is composed of two words repeated twice. Examples of double strings include pizzapizza and heyhey. Write a function called `double(s)` that returns True if and only if `s` is a double string.

Given a string `t` of length `n`, a subsequence `s` of length `m \leq n` of `t` is a string that appears in `t` when characters of `t` may be dropped. For example `ada` is a subsequence of `madman` because dropping both `ms` and the `n` from `madman` yields `ada`. Write a function called `subsequence(s, sub)` that returns True if and only if `sub` is a subsequence of `s`.
def double(s,):
    """returns True if and only if s is a double string"""
    n = len(s)
    return (n % 2 == 0) and (s[0:n//2] == s[n//2:n])

def subsequence(s,sub):
    """returns True if and only if sub is a subsequence of s"""
    start = 0
    for c in sub:
        index = s.find(c, start)
        if index == -1:
            return False
        start = index + 1
    return True