1 Review

Recall that strings are sequences so we can use the for keyword to iterate over each character in turn. For example:

```python
for c in 'purple cow':
    print(c, end=' ')
```

```plaintext
>>> purple cow
```

Let’s review indexing and slicing with some examples.

```plaintext
>>> 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'
```

2 Practice with the Python String Methods

Let’s begin by defining some new functions that directly use the Python string and sequence methods we recently introduced.

**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\t\tyob\t\tage\t\tweight".

```python
def totab(s):
    """replace the commas in 's' with tabs""
    return "\t".join(s.split(","))
```

**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’

```python
def capitalize(s):
    """return a capitalized version of s""
    return (s[0].upper + s[1:].lower())
```

**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`.

```python
def begins(s, pre):
    """returns True if and only if s begins with pre""
    return s.find(pre) == 0
```
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`

```python
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
```

Python strings support a method called `startswith` that performs that same role as `begins`. Because it is a method, the syntax is `s.startswith(pre)`. Similarly, Python strings support a method called `endswith` that performs the same role as `ends`.

3 More Practice

- 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 return `True` if and only if `s` is a double string.

```python
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])
```

- Given a string `t` of length `n`, a subsequence `s` of length `m ≤ 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`.

```python
def subsequence(s,sub):
    start = 0
    for c in sub:
        index = s.find(c, start)
        if index == -1:
            return False
        start = index + 1
    return True
```