

# CSCI 135: DIVING INTO THE DELUGE OF DATA

## LECTURE 6

strings, formatting, and sequences

# SEQUENCES

- indexing
  - $s[i]$  = the object at position  $i$
  - for strings, this yields the character at position  $i$
- slicing
  - $s[i:j:k]$  = yields a sequence of objects in the range  $s[i]$  to  $s[j-1]$  inclusive by step  $k$
  - the parameters  $i, j$ , and  $k$  are optional; for a string  $s$ 
    - $s[:4]$  = the prefix of length 4 of  $s$
    - $s[4:]$  = the suffix of  $s$  starting at position 4
    - $s[:]$  = a copy of the entire string  $s$
- length
  - $\text{len}(s)$  = the length of the sequence  $s$ ; for strings this yields the length of the string

**s = "brent drove 3.14 miles"**

s[4] =

s[1:5] =

s[13:] =

s[:5] =

## Strings are immutable

```
>>> s = "brent drove 3.14 miles"
>>> s[0] = "t"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'str' object does not support item assignment
```

# CLASSES AND METHODS

Class: Transformer

Method: transform

`optimus = Transformer()`

`optimus.transform()`



`bumble = Transformer()`

`bumble.transform()`



# METHODS ON STRINGS

- **split:** splits a string into constituent parts based on a separator string parameter
- **join:** joins a list of strings using the string object as its separating character
- **upper:** returns a copy of the string with all characters converted to upper case
- **lower:** returns a copy of the string with all characters converted to upper case
- **find:** given a search string *sub*, returns the lowest index in the string object where *sub* occurs.

```
s = "brent drove 3.14 miles"
```

```
s.split()
```

```
s.split('d')
```

```
"+".join(['3', '45', '100', '4'])
```

```
' '.join(s.split())
```

```
s.upper()
```

```
s.lower()
```

```
s.find("drove")
```

```
s.find("e", 3)
```

```
s.find("e", 12)
```

# FORMATTING STRINGS

```
>>> "{} drove {} miles".format("Brent", 3.14)
'Brent drove 3.14 miles'
```

```
>>> "{dave} drove {ten} miles".format(dave="Brent", ten=3.14)
'Brent drove 3.14 miles'
```

```
>>> "{1} drove {0} miles".format(3.14, "Brent")
'Brent drove 3.14 miles'
```



# PALINDROMES

strings that reads the same forwards and backwards

- a dog a plan a canal pagoda
- a man a plan a cat a ham a yak a  
yam a hat a canal panama
- amy must I jujitsu my ma

```
def palindrome(s):  
    n = len(s)  
    for i in range(n):  
        if (s[i] != s[n-1-i]):  
            return False  
    return True
```

```
def palindrome(s):  
    n = len(s)  
    for i in range(n//2):  
        if (s[i] != s[n-1-i]):  
            return False  
    return True
```

```
def palindrome(s):  
    return (s == s[::-1])
```

# DOUBLE

strings that, when cut in half, are the same

- brentbrent
- pizzapizza
- yeahyeah