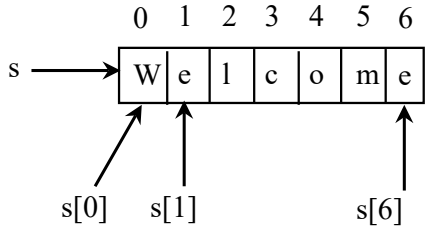
Handout 4

Strings and string methods.

- A string (type – str) is a sequence of characters, each character is also of string type.
- *String* literals can be created using matching *single quotes* (') or *double quotes* (").
  e.g. `"Good morning", 'A', '34', "56.87"`
- Some special characters:
  \n – newline \t – tab \\ - denotes \ \'– denotes ' \" - denotes "
- Python strings are **immutable**, i.e. methods and operations do not change the string (instead, they create new ones as a result)

BASIC PYTHON FUNCTIONS FOR STRINGS AND OTHER SEQUENCES




```
>>> s = "Welcome"
>>> len(s)
7
>>> s[0]
W
>>> s[0] = "w"
TypeError: 'str' object does not support item assignment
>>> s[3 : 6]  #slicing-part of the string from index 3 to index 6
'com'
>>> 'Wel' in s
True
>>> 'X' in s
False
>>> s1 = s + " to Bentley."
>>> s1
>>> "Welcome to Bentley."
>>> s2 = 2 * s
>>> s2
'WelcomeWelcome'
>>> s[-2] #negative index. Count positions from the end: len(s)-2
   'm'
>>> s[-3 : -1]
'om'
```

CHARACTERS AND NUMBERS - CONVERSION FUNCTIONS

- Python does not have a data type for characters. A single-character string represents a character. Some other languages denote a single character with a single char, hence the book follows the same convention.
- Python characters use *Unicode*, a 16-bit encoding scheme in which each symbol is numbered. That number is the *code* number of the symbol.
  Unicode is an encoding scheme for representing international characters.
  ASCII is a small subset of Unicode.

`ord(ch)` returns a number corresponding to a symbol (*Unicode*/*ASCII* table code)
`chr(num)` returns a string with the character corresponding to the num code
`str(num)` produces a string version of num

```
>>> ch = 'a'
>>> ord(ch)
 97
>>> chr(98)
 'b'
>>> s = str(3.4) # Convert a float to string
>>> s
  '3.4'
>>> s = str(3) # Convert an integer to string
>>> s
   '3'
```

PRACTICE PROBLEMS

1. Assume **Bentley course section id** is a combination of two letters, designating a subject, followed by a 3 digit course number, then section number after a dash. Given a string defining a course section, output the components separated as shown:
   Input: `cs230-006`
   Output: `Subject: CS  Course Number: 230  Section: 6`

2. Repeat the operation in 1, until user enters -1. Output how many CS courses above 100 level were entered. For example,
   ```
   Enter course section: CS213-001
   Enter course section: CS350-002
   Enter course section: CS180-001
   Enter course section: CS230-004
   Enter course section: -1
   There were 3 courses above 100 level.
   ```

3. Repeat reading course section ids as described in 1. Output a string consisting of CS courses above 100 level from the entered. For example,
   ```
   Enter course section: CS213-001
   Enter course section: CS350-002
   Enter course section: CS180-001
   Enter course section: CS230-004
   Enter course section: -1
   CS213 CS350 CS230
   ```

4. Generate a 8-char long password that includes 3 uppercase, 3 lowercase letters and two digits. *Hint*: Produce a random character using random integer generator. To generate a random integer **num** in the range from number 5 to number 10, inclusively, include the following code:
   ```
   import random #put in the top of the program
   num = random.randint(5, 11)
   ```

STRING METHODS

Method – a function that is **called by an object.** In the following, s is the calling object, where upper is the method

```
>>> s = "Welcome"
>>> s.upper()
 'WELCOME'
```

ON the other hand, len() is a function, not a method, because it does not require a calling object.

- Recall, that Python strings are **immutable**, i.e. methods and operations do not change the string (instead, they create new ones as a result).

What is the value of s after the above segment has been executed?

Modifying and formatting– the titles of the methods are pretty self-explanatory

`capitalize()`	`lstrip()`
`lower()`	`rstrip()`
`upper()`	`strip()`
`title()`	`center(width)`
`swapcase()`	`ljust(width)`
`replace(old, new)`	`rjust(width)`
	`format(items)`

Testing characters in a function – return a boolean value True or False

Assuming s is an object of type str,
method returns `True` iff **s has at least one character and**

`s.isalnum()`	all characters in s are **alphanumeric**
`s.isalpha()`	all characters in s are **alphabetic**
`s.isdigit()`	s contains only **number** characters.
`s.islower()`	s contains only **lowercase** letters.
`s.isupper()`	s contains only **uppercase** letters.
`s.isspace()`	**s** contains only **whitespace** characters (newlines, spaces, tabs, etc).

PRACTICE PROBLEMS

5. Check if a word entered by a user defines a phone number, e.g.

```
213-001-2345
is a phone number
```

or

```
CIS-210001
Is not a phone number
```

6. User will enter a string. Print how many letters, how many digits, spaces and other symbols are in it. Hint: examine it character-by-character using a loop
7. User will enter a string. Replace all punctuation in it.
   a. Simpler version: Assume punctuation symbols are `,.:;!?.` Are you using a loop to do it ?

b. Extra challenge: Assume a punctuation symbol is anything that is not a letter, a digit, or a space. *Hint*: first, go through the string, examining every character and collecting a string of punctuation symbols. After that, in a loop, replace every punctuation symbol.

Searching for Substrings.

Assuming s and s1 are strings. [ ] mean parameter is optional

`s.endswith(s1)`	returns True if s **ends** with s1
`s.startswith(s1)`	returns True if the string **starts** with s1
`s.find(s1 [,start[,end]])`	Returns the **lowest** index where s1 starts in this string, between positions **`start`** and **`end`**, or -1 if s1 is not found in this string.
`s.rfind(s1 [,start[,end]])`	Returns the **highest** index where s1 starts in this string between positions **`start`** and **`end`**, or -1 if s1 is not found in this string.
`s.count(s1)`	Returns the **number of non-overlapping occurrences** of s1

PRACTICE PROBLEMS

8. User will be entering email addresses one per line, followed by word STOP. For each entry, separate the username from the domain. Print "Bentley" if the email ends with bentely.edu
9. User will enter string defining distance in either inches or feet until -1 is entered. Produce the total length in inches, e.g.

```
Enter next value: 3 feet
Enter next value: 0.5 Feet
Enter next value: 5 inches
Enter next value: 2 FEET
Enter next value: -1
Total 71 inches
```

Extra challenges:

a) assume there may be any number of spaces separating the numeric value from the string

b) allow both feet and inches to be entered on one line, e.g.

```
3 feet 2 inches
```

or, even

```
3 inches   7 feet
```

Extra -- Method split()

`s.split(sep=None, maxsplit=-1)`

`sep – optional parameter, separator between the words`

`maxsplit – optional parameter – number of seps considered`

Return a list of words (***word*** *is a sequence of characters not equal to* ***sep***) in string s, separated by sep.

If separator is not specified, **all runs of consecutive whitespace are regarded as a single separator.**

If **`maxsplit`** is given, at most maxsplit splits are done (i.e. consider only the first **`maxsplit`** separators, thus, the list will have at most maxsplit+1 elements). If **`maxsplit`** is not specified or -1, then there is no limit on the number of splits (all possible splits are made).

```
>>>words = "Welcome  to    the  US\n".split()
>>>words
       ['Welcome', 'to', 'the', 'US']
>>>words[0]
      'Welcome'
>>>words[1]
      'to'
>>>words[-1]
      'US'
>>>"34-13-foo-45".split("-")  # - used as a separator
      ['34', '13', 'foo', '45']
```