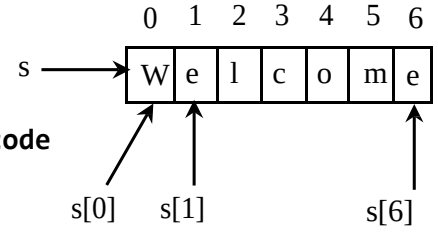
Handout 2

Strings and string methods.

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

BASIC STRING FUNCTIONS




```
>>> s = "Welcome"
>>> len(s)
7
>>> max(s) # min char by code
o
>>> min(s)
W
>>> s[0]
W
>>> s[3 : 6]  #slicing-part of the string from index 3 to index 6
'com'
>>> 'Wel' in s
True
>>> 'X' in s
False
>>> s2 = 2 * s
>>> s2
'WelcomeWelcome'
>>> s[-2] #negative index. Count positions from the end: len(s)-2
   'm'
>>> s[-3 : -1]
'om'
```

CONVERSION FUNCTIONS

`ord(ch)` returns a character corresponding to a number (based on *Unicode/ASCII* tables)

`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'
```

STRING METHODS

Method – a function that is **called by an object** (see also Handout 1), e.g. in the following, s is the calling object, where upper is the method

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

Recall, that the string methods do not change the calling string.

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

Method split() – separating string components

`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'

>>>""Welcome  to    the  US\n"".split(' ')
['Welcome', '', 'to', '', '', '', 'the', '', 'US\n'

>>>"34-13-foo-45".split("-")  # - used as a separator
      ['34', '13', 'foo', '45']
>>>"34-13-foo-45".split("-", 2)
      ['34', '13', 'foo-45']

>>>> 'aaa'.split('a')
      ['', '', '', '']
>>>  'aaa'.split('aa')
      ['', 'a']

>>> '1,2,3'.split(',', maxsplit=1)
['1', '2,3']
>>> '1 2 3'.split(maxsplit=1)
   ['1', '2 3']
```

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).

Searching for Substrings.

Assuming s and s1 are strings

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

Converting 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)

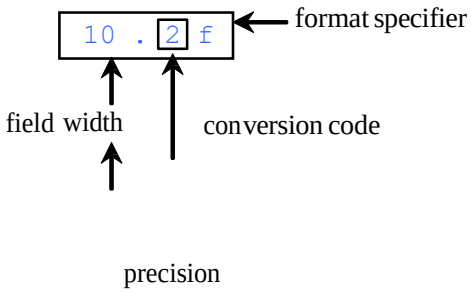
FORMATTING

format(item, format-specifier) returns a string with **item** formatted according to **format-specifier**

item is a number or a string, format-specifier is a string using special formatting instructions. See the book or documentation for complete set.

Formatting symbols and conversion codes:
`f` float `s` string `d` decimal int `x` hexadecimal int `b` binary int `%` percentage
< left-justified > right-justified

```
print(format(57.467657, '10.2f'))
print(format(12345678.923,  '10.2f'))
print(format(57.4, '10.2f'))
print(format(57, '10.2f'))
```




```
 ← 10 →
□□□□□57.47
 12345678.92
□□□□□57.40
□□□□□57.00
```

PRACTICE PROBLEMS ON STRINGS

1. Given a string that contains a phone number without dashes, e.g. "7813456789", compose another string that has the phone with the dashes: "781-345-6789".
2. Given a string, compose another one, which contains the first character and the last character, in uppercase. For example, for string "Bentley", the resulting string should be "BY".
3. Given a string of even length, produce the second half. So the string "WooHoo" yields "Hoo".
4. Given a string containing a sentence with parentheses, print out the test inside parentheses, removing all surrounding spaces; for example, given "There was snow (a lot of it!) last week." Output "a lot of it!"
5. Given a string of text, replace all space characters with dashes, e.g. for "There was snow" the output should be "There-was-snow".
6. Given a word that contains letter a, e.g. "blackboard" produce one that has two letters after the first 'a' capitalized, i.e. "blaCKboard"
7. Given a string of text with several words, e.g. "brevity is the soul of wit" change it so that it has
    a. the first word capitalized, i.e. "BREVITY is the soul of wit",
    b. the last word capitalized, i.e. "brevity is the soul of WIT"
8. Eliminate all linebreaks in text.