Strings

• We have already seen strings – they are sequences of characters delimited by ' and ' or " "

• Let's take a closer look…
String literals

- A string literal is anything in quotes
- But everything in the computer is stored in binary, so each character is stored as a number
- Examples:
  
  - `ord("a") -> 97`
  - `chr(97) -> a`
  - `ord("b") -> 98`
  - `ord("A") -> 65`
  - "a" < "A" -> False
## ASCII values

### ASCII Table

<table>
<thead>
<tr>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
<th>Decimal</th>
<th>Hex</th>
<th>Char</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>[NULL]</td>
<td>32</td>
<td>20</td>
<td>[SPACE]</td>
<td>64</td>
<td>40</td>
<td>@</td>
<td>96</td>
<td>60</td>
<td>`</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>[START OF HEADING]</td>
<td>33</td>
<td>21</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>A</td>
<td>97</td>
<td>61</td>
<td>a</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>[START OF TEXT]</td>
<td>34</td>
<td>22</td>
<td>&quot;</td>
<td>66</td>
<td>42</td>
<td>B</td>
<td>98</td>
<td>62</td>
<td>b</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>[END OF TEXT]</td>
<td>35</td>
<td>23</td>
<td>#</td>
<td>67</td>
<td>43</td>
<td>C</td>
<td>99</td>
<td>63</td>
<td>c</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>[END OF TRANSMISSION]</td>
<td>36</td>
<td>24</td>
<td>$</td>
<td>68</td>
<td>44</td>
<td>D</td>
<td>100</td>
<td>64</td>
<td>d</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>[ENQUIRY]</td>
<td>37</td>
<td>25</td>
<td>%</td>
<td>69</td>
<td>45</td>
<td>E</td>
<td>101</td>
<td>65</td>
<td>e</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>[ACKNOWLEDGE]</td>
<td>38</td>
<td>26</td>
<td>&amp;</td>
<td>70</td>
<td>46</td>
<td>F</td>
<td>102</td>
<td>66</td>
<td>f</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>[BEL]</td>
<td>39</td>
<td>27</td>
<td>'</td>
<td>71</td>
<td>47</td>
<td>G</td>
<td>103</td>
<td>67</td>
<td>g</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>[BACKSPACE]</td>
<td>40</td>
<td>28</td>
<td>(</td>
<td>72</td>
<td>48</td>
<td>H</td>
<td>104</td>
<td>68</td>
<td>h</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>[HORIZONTAL TAB]</td>
<td>41</td>
<td>29</td>
<td>)</td>
<td>73</td>
<td>49</td>
<td>I</td>
<td>105</td>
<td>69</td>
<td>i</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>[VERTICAL TAB]</td>
<td>43</td>
<td>2B</td>
<td>+</td>
<td>75</td>
<td>4B</td>
<td>K</td>
<td>107</td>
<td>6B</td>
<td>k</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>[FORM FEED]</td>
<td>44</td>
<td>2C</td>
<td>-</td>
<td>76</td>
<td>4C</td>
<td>L</td>
<td>108</td>
<td>6C</td>
<td>l</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>[CARRIAGE RETURN]</td>
<td>45</td>
<td>2D</td>
<td>.</td>
<td>77</td>
<td>4D</td>
<td>M</td>
<td>109</td>
<td>6D</td>
<td>m</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>[SHIFT OUT]</td>
<td>46</td>
<td>2E</td>
<td>\</td>
<td>78</td>
<td>4E</td>
<td>N</td>
<td>110</td>
<td>6E</td>
<td>n</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>[SHIFT IN]</td>
<td>47</td>
<td>2F</td>
<td>/</td>
<td>79</td>
<td>4F</td>
<td>O</td>
<td>111</td>
<td>6F</td>
<td>o</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>[DATA LINK ESCAPE]</td>
<td>48</td>
<td>30</td>
<td>0</td>
<td>80</td>
<td>50</td>
<td>P</td>
<td>112</td>
<td>70</td>
<td>p</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>[DEVICE CONTROL 1]</td>
<td>49</td>
<td>31</td>
<td>1</td>
<td>81</td>
<td>51</td>
<td>Q</td>
<td>113</td>
<td>71</td>
<td>q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>[DEVICE CONTROL 2]</td>
<td>50</td>
<td>32</td>
<td>2</td>
<td>82</td>
<td>52</td>
<td>R</td>
<td>114</td>
<td>72</td>
<td>r</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>[DEVICE CONTROL 3]</td>
<td>51</td>
<td>33</td>
<td>3</td>
<td>83</td>
<td>53</td>
<td>S</td>
<td>115</td>
<td>73</td>
<td>s</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>[DEVICE CONTROL 4]</td>
<td>52</td>
<td>34</td>
<td>4</td>
<td>84</td>
<td>54</td>
<td>T</td>
<td>116</td>
<td>74</td>
<td>t</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>[NEGATIVE ACKNOWLEDGE]</td>
<td>53</td>
<td>35</td>
<td>5</td>
<td>85</td>
<td>55</td>
<td>U</td>
<td>117</td>
<td>75</td>
<td>u</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>[SYNCHRONOUS IDLE]</td>
<td>54</td>
<td>36</td>
<td>6</td>
<td>86</td>
<td>56</td>
<td>V</td>
<td>118</td>
<td>76</td>
<td>v</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>[ENQ OF TRANS. BLOCK]</td>
<td>55</td>
<td>37</td>
<td>7</td>
<td>87</td>
<td>57</td>
<td>W</td>
<td>119</td>
<td>77</td>
<td>w</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>[CANCEL]</td>
<td>56</td>
<td>38</td>
<td>8</td>
<td>88</td>
<td>58</td>
<td>X</td>
<td>120</td>
<td>78</td>
<td>x</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>[END OF MEDIUM]</td>
<td>57</td>
<td>39</td>
<td>9</td>
<td>89</td>
<td>59</td>
<td>Y</td>
<td>121</td>
<td>79</td>
<td>y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>[SUBSTITUTE]</td>
<td>58</td>
<td>3A</td>
<td>:</td>
<td>90</td>
<td>5A</td>
<td>Z</td>
<td>122</td>
<td>7A</td>
<td>z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>[ESCAPE]</td>
<td>59</td>
<td>3B</td>
<td>;</td>
<td>91</td>
<td>5B</td>
<td>[</td>
<td>123</td>
<td>7B</td>
<td>{</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>[FILE SEPARATOR]</td>
<td>60</td>
<td>3C</td>
<td>&lt;</td>
<td>92</td>
<td>5C</td>
<td>\</td>
<td>124</td>
<td>7C</td>
<td></td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>[GROUP SEPARATOR]</td>
<td>61</td>
<td>3D</td>
<td>=</td>
<td>93</td>
<td>5D</td>
<td>J</td>
<td>125</td>
<td>7D</td>
<td>}</td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>[RECORD SEPARATOR]</td>
<td>62</td>
<td>3E</td>
<td>&gt;</td>
<td>94</td>
<td>5E</td>
<td>^</td>
<td>126</td>
<td>7E</td>
<td>~</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>[UNIT SEPARATOR]</td>
<td>63</td>
<td>3F</td>
<td>?</td>
<td>95</td>
<td>5F</td>
<td>[DEL]</td>
<td>127</td>
<td>7F</td>
<td>[DEL]</td>
</tr>
</tbody>
</table>
def toUpperCaseLetter(character):
    if ("a" <= character <= "z"):
        return chr(ord(character) - 32)
    return character
Escape sequences

• Escape sequences:
  – single quote \'
  – double quote \"
  – backslash \\n  – newline \n
  – tab \t
String operators

• Operators:
  – Concatenation  +
  – Multiple concatenation  *
  – Length  len (a function)
  – Indexing [valid values are -len(s) to len(s) -1]
    • <string>[n]
      – gives you the character at position n (starting from 0)
    • <string>[-n]
      – gives you the character at position len(<string>)-n
    • examples…
String Indexing

s = "Professor Mark"

len(s) -> 14 (so valid indices are -14 .. 13)
s[0] -> P
s[len(s)-1] -> k
s[-1] -> k
s[-14] -> P
s[42] -> error
More string operators

- Slicing
  - `<string>[start:end:step]`
    - gives you the substring beginning at start up to but not including end
  - Examples
    - `s = "Professor Mark"
    - `s[10:12] -> Ma`
    - `s[10:] -> Mark`
    - `s[10:] -> Professor (with the space)`
More string operators

– Contains
  • in
    – "ark" in "Mark" -> True
    – "Mark" in "Professor Mark" -> True
    – "Mark" in "Professor" -> False
  • not in (this is OK in Python, as opposed to not (c in s))
    – not "Mark" in "Professor" -> True
    – "Mark" not in "Professor" -> True
Strings are immutable

• A string, once created, cannot be modified
  
  s = "abcd"
  s[0] = "d"  # error!

• But s can hold a different, new string…
  
  s += "efg"
  print(s)  # prints "abcdefg"  Why?

  Suppose I wanted to reverse the contents of a string variable? How could I do that?
Strings and loops

• Iterating over a string with a for loop
  – likely to use len()
  – an example
    for i in range(len(<string-variable>)):
      print(i, s[i])
  – a different way to iterate over a string (if position is not needed):
    for c in <string-variable>:
      print(c)
  – examples: let's write isInteger() and isPalindrome()
String constants

- String constants (must do what to use these?):
  - string.ascii_letters 'a..zA..Z'
  - string.ascii_lowercase 'a..z'
  - string.ascii_uppercase 'A..Z'
  - string.digits '0123456789'
  - string.punctuation lots of things 😊
  - string.whitespace space, tab, return
  - string.printable letters + digits + punc + whitsp
String methods (v. functions, constants)

- String functions and methods
  - Functions take a string as a parameter, e.g.,
    - `len()` – takes what as a parameter? returns what?
    - `input()` – takes what as a parameter? returns what?
  - Methods operate on a particular string, e.g.,
    - `<str>.find()` [and `<str>.replace()`, `<str>.count()`]
    - `<str>.isdigit()` [`.isalpha()`, `.islower()`, `.isupper()`, `.isspace()`]
    - `<str>.lower()` [and `<str>.upper()`, `<str>.capitalize()`]
    - `<str>.split()` [and `<str>.strip()`]
    - https://docs.python.org/3/library/stdtypes.html?highlight=strip#string-methods
String Formatting

• How to create a *formatting* string, which looks like:
  – "format_string" % (values)
  – The *format_string* contains conversion specifiers
    • %s – string
    • %d – integer
    • %f – floating point
    • %c – single character
    • %% – the character '%'
  – Specifiers can be preceded by optional width.precision
String Formatting

• If there is more than one conversion specifier, (values) must have the same number of items, and is called a *tuple*

• For each conversion specifier (%x) in the format string, there must be a corresponding value in the tuple of values

• Examples:
  - \( s = \text{last} + ", " + \text{first} \)
  - \( s2 = \"%s, %s\" \% (\text{last}, \text{first}) \)
  - \( s3 = \"%s\t%s\" \% (\text{first}, \text{last}) \)
  - \( s4 = \"|%s|%s|%s|\" \% (\"x\", \"o\", \"x\") \)
  - \( s5 = \"\text{Pi is approximately \%0.2f\} \% (22/7) \)