Strings

- Strings are actually 'objects'
What is an object?!

- An object is a concept that we can encapsulate data along with the functions that might need to access or manipulate that data.
What is an object?!

- An object is a concept that we can encapsulate data along with the functions that might need to access or manipulate that data.
- Every string we create is a new string object composed of:
What is an object?!

• An object is a concept that we can encapsulate data along with the functions that might need to access or manipulate that data.

• Every string we create is a new string object composed of:
  • the letters/numbers/etc. in that string
What is an object?!

- An object is a concept that we can encapsulate data along with the functions that might need to access or manipulate that data.

- Every string we create is a new string object composed of:
  - the letters/numbers/etc. in that string
  - a set of functions that can be used to access/manipulate that string
Strings and Characters

- Strings are composed of a set of characters
- New data type: characters ('chr' in python)
- A character is a single letter, number, symbol, or even non-printable characters (backspace for example)
Escape Characters

Some characters have special meanings in python. For example:

' # single quote
"
" # double quote
Escape Characters

To reference these characters we use the 'escape' character to interpret them:
\
Putting the escape character in front of certain characters provides a special interpretation for python to treat that character differently.
Common Escaped Characters

```
\  # tells python to treat back slash as a backslash
print('The following is a backslash: \\

\
  # newline
print('Joe''s a programmer:')

'' # tells python to treat ' as a literal double quote
print('''Joe said, ''I like python.'''

\n  # newline
\t  # tab
```
Character Values

Characters have values... based on the ASCII/Unicode character encoding.
## Character ASCII/Unicode Values

<table>
<thead>
<tr>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Char</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
<th>Dec</th>
<th>Hx</th>
<th>Oct</th>
<th>Html</th>
<th>Chr</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>000</td>
<td>NUL (null)</td>
<td>32</td>
<td>20</td>
<td>040</td>
<td>…</td>
<td>Space</td>
<td>64</td>
<td>40</td>
<td>100</td>
<td>…</td>
<td>96</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>001</td>
<td>SOH (start of heading)</td>
<td>33</td>
<td>21</td>
<td>041</td>
<td>…</td>
<td>!</td>
<td>65</td>
<td>41</td>
<td>101</td>
<td>…</td>
<td>A</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>002</td>
<td>STX (start of text)</td>
<td>34</td>
<td>22</td>
<td>042</td>
<td>…</td>
<td>!</td>
<td>66</td>
<td>42</td>
<td>102</td>
<td>…</td>
<td>B</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>003</td>
<td>ETX (end of text)</td>
<td>35</td>
<td>23</td>
<td>043</td>
<td>…</td>
<td>!</td>
<td>67</td>
<td>43</td>
<td>103</td>
<td>…</td>
<td>C</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>004</td>
<td>EOT (end of transmission)</td>
<td>36</td>
<td>24</td>
<td>044</td>
<td>…</td>
<td>!</td>
<td>68</td>
<td>44</td>
<td>104</td>
<td>…</td>
<td>D</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>005</td>
<td>ENQ (enquiry)</td>
<td>37</td>
<td>25</td>
<td>045</td>
<td>…</td>
<td>!</td>
<td>69</td>
<td>45</td>
<td>105</td>
<td>…</td>
<td>E</td>
</tr>
<tr>
<td>6</td>
<td>6</td>
<td>006</td>
<td>ACK (acknowledge)</td>
<td>38</td>
<td>26</td>
<td>046</td>
<td>…</td>
<td>!</td>
<td>70</td>
<td>46</td>
<td>106</td>
<td>…</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>007</td>
<td>BEL (bell)</td>
<td>39</td>
<td>27</td>
<td>047</td>
<td>…</td>
<td>!</td>
<td>71</td>
<td>47</td>
<td>107</td>
<td>…</td>
<td>G</td>
</tr>
<tr>
<td>8</td>
<td>8</td>
<td>010</td>
<td>BS (backspace)</td>
<td>40</td>
<td>28</td>
<td>050</td>
<td>…</td>
<td>!</td>
<td>72</td>
<td>48</td>
<td>108</td>
<td>…</td>
<td>H</td>
</tr>
<tr>
<td>9</td>
<td>9</td>
<td>011</td>
<td>TAB (horizontal tab)</td>
<td>41</td>
<td>29</td>
<td>051</td>
<td>…</td>
<td>!</td>
<td>73</td>
<td>49</td>
<td>109</td>
<td>…</td>
<td>I</td>
</tr>
<tr>
<td>10</td>
<td>A</td>
<td>012</td>
<td>LF (NL line feed, new line)</td>
<td>42</td>
<td>2A</td>
<td>052</td>
<td>…</td>
<td>!</td>
<td>74</td>
<td>4A</td>
<td>112</td>
<td>…</td>
<td>J</td>
</tr>
<tr>
<td>11</td>
<td>B</td>
<td>013</td>
<td>VT (vertical tab)</td>
<td>43</td>
<td>2B</td>
<td>053</td>
<td>…</td>
<td>!</td>
<td>75</td>
<td>4B</td>
<td>113</td>
<td>…</td>
<td>K</td>
</tr>
<tr>
<td>12</td>
<td>C</td>
<td>014</td>
<td>FF (NP form feed, new page)</td>
<td>44</td>
<td>2C</td>
<td>054</td>
<td>…</td>
<td>!</td>
<td>76</td>
<td>4C</td>
<td>114</td>
<td>…</td>
<td>L</td>
</tr>
<tr>
<td>13</td>
<td>D</td>
<td>015</td>
<td>CR (carriage return)</td>
<td>45</td>
<td>2D</td>
<td>055</td>
<td>…</td>
<td>!</td>
<td>77</td>
<td>4D</td>
<td>115</td>
<td>…</td>
<td>M</td>
</tr>
<tr>
<td>14</td>
<td>E</td>
<td>016</td>
<td>SO (shift out)</td>
<td>46</td>
<td>2E</td>
<td>056</td>
<td>…</td>
<td>!</td>
<td>78</td>
<td>4E</td>
<td>116</td>
<td>…</td>
<td>N</td>
</tr>
<tr>
<td>15</td>
<td>F</td>
<td>017</td>
<td>SI (shift in)</td>
<td>47</td>
<td>2F</td>
<td>057</td>
<td>…</td>
<td>!</td>
<td>79</td>
<td>4F</td>
<td>117</td>
<td>…</td>
<td>O</td>
</tr>
<tr>
<td>16</td>
<td>10</td>
<td>020</td>
<td>DLE (data link escape)</td>
<td>48</td>
<td>30</td>
<td>060</td>
<td>…</td>
<td>!</td>
<td>80</td>
<td>50</td>
<td>120</td>
<td>…</td>
<td>P</td>
</tr>
<tr>
<td>17</td>
<td>11</td>
<td>021</td>
<td>DC1 (device control 1)</td>
<td>49</td>
<td>31</td>
<td>061</td>
<td>…</td>
<td>!</td>
<td>81</td>
<td>51</td>
<td>121</td>
<td>…</td>
<td>Q</td>
</tr>
<tr>
<td>18</td>
<td>12</td>
<td>022</td>
<td>DC2 (device control 2)</td>
<td>50</td>
<td>32</td>
<td>062</td>
<td>…</td>
<td>!</td>
<td>82</td>
<td>52</td>
<td>122</td>
<td>…</td>
<td>R</td>
</tr>
<tr>
<td>19</td>
<td>13</td>
<td>023</td>
<td>DC3 (device control 3)</td>
<td>51</td>
<td>33</td>
<td>063</td>
<td>…</td>
<td>!</td>
<td>83</td>
<td>53</td>
<td>123</td>
<td>…</td>
<td>S</td>
</tr>
<tr>
<td>20</td>
<td>14</td>
<td>024</td>
<td>DC4 (device control 4)</td>
<td>52</td>
<td>34</td>
<td>064</td>
<td>…</td>
<td>!</td>
<td>84</td>
<td>54</td>
<td>124</td>
<td>…</td>
<td>T</td>
</tr>
<tr>
<td>21</td>
<td>15</td>
<td>025</td>
<td>NAK (negative acknowledge)</td>
<td>53</td>
<td>35</td>
<td>065</td>
<td>…</td>
<td>!</td>
<td>85</td>
<td>55</td>
<td>125</td>
<td>…</td>
<td>U</td>
</tr>
<tr>
<td>22</td>
<td>16</td>
<td>026</td>
<td>SYN (synchronous idle)</td>
<td>54</td>
<td>36</td>
<td>066</td>
<td>…</td>
<td>!</td>
<td>86</td>
<td>56</td>
<td>126</td>
<td>…</td>
<td>V</td>
</tr>
<tr>
<td>23</td>
<td>17</td>
<td>027</td>
<td>ETB (end of trans. block)</td>
<td>55</td>
<td>37</td>
<td>067</td>
<td>…</td>
<td>!</td>
<td>87</td>
<td>57</td>
<td>127</td>
<td>…</td>
<td>W</td>
</tr>
<tr>
<td>24</td>
<td>18</td>
<td>030</td>
<td>CAN (cancel)</td>
<td>56</td>
<td>38</td>
<td>070</td>
<td>…</td>
<td>!</td>
<td>88</td>
<td>58</td>
<td>130</td>
<td>…</td>
<td>X</td>
</tr>
<tr>
<td>25</td>
<td>19</td>
<td>031</td>
<td>EM (end of medium)</td>
<td>57</td>
<td>39</td>
<td>071</td>
<td>…</td>
<td>!</td>
<td>89</td>
<td>59</td>
<td>131</td>
<td>…</td>
<td>Y</td>
</tr>
<tr>
<td>26</td>
<td>1A</td>
<td>032</td>
<td>SUB (substitute)</td>
<td>58</td>
<td>3A</td>
<td>072</td>
<td>…</td>
<td>!</td>
<td>90</td>
<td>5A</td>
<td>132</td>
<td>…</td>
<td>Z</td>
</tr>
<tr>
<td>27</td>
<td>1B</td>
<td>033</td>
<td>ESC (escape)</td>
<td>59</td>
<td>3B</td>
<td>073</td>
<td>…</td>
<td>!</td>
<td>91</td>
<td>5B</td>
<td>133</td>
<td>…</td>
<td>{</td>
</tr>
<tr>
<td>28</td>
<td>1C</td>
<td>034</td>
<td>FS (file separator)</td>
<td>60</td>
<td>3C</td>
<td>074</td>
<td>…</td>
<td>!</td>
<td>92</td>
<td>5C</td>
<td>134</td>
<td>…</td>
<td>\</td>
</tr>
<tr>
<td>29</td>
<td>1D</td>
<td>035</td>
<td>GS (group separator)</td>
<td>61</td>
<td>3D</td>
<td>075</td>
<td>…</td>
<td>!</td>
<td>93</td>
<td>5D</td>
<td>135</td>
<td>…</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>1E</td>
<td>036</td>
<td>RS (record separator)</td>
<td>62</td>
<td>3E</td>
<td>076</td>
<td>…</td>
<td>!</td>
<td>94</td>
<td>5E</td>
<td>136</td>
<td>…</td>
<td>^</td>
</tr>
<tr>
<td>31</td>
<td>1F</td>
<td>037</td>
<td>US (unit separator)</td>
<td>63</td>
<td>3F</td>
<td>077</td>
<td>…</td>
<td>!</td>
<td>95</td>
<td>5F</td>
<td>137</td>
<td>…</td>
<td>_</td>
</tr>
</tbody>
</table>

Source: www.LookupTables.com
Character Conversions

- `chr(i)` - returns the character for the unicode value passed to it
  - `chr(97)` returns 'a' (based on ASCII chart)
  - `chr(65)` returns 'A'
  - `chr(49)` returns '1'
- `ord(c)` - returns the unicode value (int) for character 'c' passed to it
  - `ord('z')` returns 122 (based on ASCII chart)
  - `ord('Z')` returns 90
  - `ord('9')` returns 57
String Indexing

mystring = 'python'
len(mystring) # returns 6

mystring[0] == 'p'               mystring[-1] == 'n'
mystring[1] == 'y'               mystring[-2] == 'o'
...                              ...

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>mystring</td>
<td>p</td>
<td>y</td>
<td>t</td>
<td>h</td>
<td>o</td>
<td>n</td>
</tr>
<tr>
<td>positive index</td>
<td>[0]</td>
<td>[1]</td>
<td>[2]</td>
<td>[3]</td>
<td>[4]</td>
<td>[5]</td>
</tr>
</tbody>
</table>
String Slicing

mystring = 'python'
len(mystring) # returns 6
mystring[:] # returns 'python'
mystring[1:4] # returns 'yth'
mystring[:6] # returns 'python'
mystring[:5] # returns 'pytho'

# Slicing is always increasing..
mystring[:-5] # returns 'p'
mystring[:6] # returns "
mystring[-3:-5] # returns "
mystring[-5:-3] # returns 'yt'
String Functions

- help(str) for a full list of string functions (or see the python website)
- String functions return a new string!
- String functions do not modify the original string!
String Functions

Some helpful functions:

- capitalize(...)  
  - S.capitalize() -> str  
  - Return a capitalized version of S, i.e. make the first character have upper case.
  - Example:
    >>> mystring = 'python'
    >>> mystring.capitalize()
    'Python'
    >>> mystring
    'python'
    >>> newstring = mystring.capitalize()
    >>> newstring
    'Python'
String Functions (cont'd)

• More helpful functions:
  • title(…)
    - S.title() -> str
    - Return a titlecased version of S, i.e. words start with title case characters, all remaining cased characters have lower case.
    - Example:
      >>> 'python is cool'.title()
      'Python Is Cool'
Some helpful functions:

- `startswith(...)`
  - `S.startswith(prefix[, start[, end]])` -> bool
  - Return True if `S` starts with the specified prefix, False otherwise. With optional start, test `S` beginning at that position. With optional end, stop comparing `S` at that position. `prefix` can also be a tuple of strings to try.
  - Example:
    ```python
test = 'python'.startswith('py')
print(test)
```
    ```python
True
```
Some helpful functions:

- `endswith(...)`
  - `S.endswith(suffix[, start[, end]])` -> bool
  - Return True if S ends with the specified suffix, False otherwise. With optional start, test S beginning at that position. With optional end, stop comparing S at that position. suffix can also be a tuple of strings to try.
  - Example:
    ```python
    >>> 'python'.endswith('on')
    True
    ```
String Functions

• Some helpful functions:
  • find(...)
    - `S.find(sub[, start[, end]])` -> int
    - Return the lowest index in S where substring sub is found, such that sub is contained within `s[start:end]`. Optional arguments start and end are interpreted as in slice notation.
    - Example:
      ```python
      >>> '123python123'.find('2')
      1
      ```
String Functions (cont'd)

- More helpful functions:
  - `rfind(...)`
    - `S.rfind(sub[, start[, end]]) -> int`
    - Return the highest index in S where substring sub is found, such that sub is contained within `s[start:end]`. Optional arguments start and end are interpreted as in slice notation.
    - Example:
      ```
      >>> '123python123'.rfind('2')
      10
      ```
String Functions (cont'd)

• More helpful functions:
  • isalpha(...)  
    - S.isalpha() -> bool  
    - Return True if all characters in S are alphabetic and there is at least one character in S, False otherwise.  
    - Example:
      >>> 'python'.isalpha()
      True
      >>> 'python123'.isalpha()
      False
String Functions (cont'd)

• More helpful functions:
  • isdigit(...)  
    - S.isdigit() -> bool  
    - Return True if all characters in S are digits and there is at least one character in S, False otherwise  
    - Example:

      >>> 'python'.isdigit()
      False
      >>> '12345'.isdigit()
      True
      >>> 'py123'.isdigit()
      False
String Functions (cont'd)

- More helpful functions:
  - islower(...)
    - S.islower() -> bool
    - Return True if all cased characters in S are lowercase and there is at least one cased character in S, False otherwise.
    - Example:
      >>> 'python'.islower()
      True
      >>> 'Python'.islower()
      False
String Functions (cont'd)

• More helpful functions:
  • isupper(...) 
    - S.isupper() -> bool
    - Return True if all cased characters in S are uppercase and there is at least one cased character in S, False otherwise.
    - Example:
      >>> 'Python'.isupper()
      False
      >>> 'PYTHON'.isupper()
      True
String Functions (cont'd)

- More helpful functions:
  - `isspace(...)`
    - `S.isspace() -> bool`
    - Return True if all characters in S are whitespace and there is at least one character in S, False otherwise.
    - Example:
      ```python
      >>> 'python'.isspace()
      False
      >>> '
      True
      ```
String Functions (cont'd)

- More helpful functions:
  - `upper(...)`
    - `S.upper()` -> str
    - Return a copy of S converted to uppercase.
    - Example:
      ```python
      >>> 'Python'.upper()
      'PYTHON'
      >>> python'.upper()
      'PYTHON'
      ```
String Functions (cont'd)

- More helpful functions:
  - `lower(...)
    - `S.lower()` -> str
    - Return a copy of the string S converted to lowercase.
    - Example:
      
      ```
      >>> 'Python'.lower()
      'python'
      >>> 'PYTHON'.lower()
      'python'
      ```
More helpful functions:

- `split(...)`
  - `S.split([sep[, maxsplit]]) -> list of strings`
  - Return a list of the words in `S`, using `sep` as the delimiter string. If `maxsplit` is given, at most `maxsplit` splits are done. If `sep` is not specified or is `None`, any whitespace string is a separator and empty strings are removed from the result.

- Example:
  ```python
  >>> 'abc,def,ghi'.split(',')
  ['abc', 'def', 'ghi']
  ```
More helpful functions:

- `strip(...)`
  - `S.strip([chars]) -> str`
  - Return a copy of the string `S` with leading and trailing whitespace removed.
  - Example:
    ```python
    >>> '  python  '.strip()
    'python'
    ```
String Functions (cont'd)

• More helpful functions:
  • replace(…)
    – S.replace (old, new[, count]) -> str
    – Return a copy of S with all occurrences of substring old replaced by new. If the optional argument count is given, only the first count occurrences are replaced.
    – Example:
      
      >>> 'python is cool'.replace('cool','fun')
      'python is fun'