Strings

Announcements *(I usually write these on the board)*
- HW #3 due tomorrow at 11pm
- Exam #2 Tuesday 7-8:30pm
- No office hours today -- email or attend open lab hours

Outline
- Warm-up
- String indexing and slicing
- String methods
- String comparison

Warm-up *(Students start on this when they come into class and work for the first few minutes of class)*

*What is the output of this program?*

```python
def func1(a, b):
    print(a * b)
    return a // b

def func2(x):
    x = x * 2

def func3(x):
    print(x ** 2)

def main():
    i = 4
    j = func1(i, 2)
    func2(i)
    print(i)
    x = 5
    func2(x)
    print(x)
    func3(j)
    print(func3(j))

main()
```

*Output:*
8
4
5
4
4
None
Strings

- Already know how to...
  - Represent a string --- surround sequence of characters with single ('), double ("), triple-single ('''), or triple-double (""") quotes
  - Concatenate strings --- using + operator
  - Repeat strings --- using * operator
  - Count the number of characters --- using len function

String indexing and slicing

- Index operator --- used to select a single character in a string
  - Characters are numbered left to right starting from 0
    - Example:
      ```python
      myString = 'Colgate 13'
      print(myString[0])  # Outputs C
      print(myString[4])  # Outputs a
      ```
  - Can also use negative numbers (starting at -1) to get a character relative to the end of the string
    - Example
      ```python
      print(myString[-1])  # Outputs 3
      print(myString[-10])  # Outputs C
      ```

- Slice operator --- used to select a contiguous subset of characters in a string
  - Put starting and ending indices in square brackets separated by a colon
    - Includes character at starting index, but excludes character at ending index --- similar to sequence generated by range going to STOP-1
    - Example
      ```python
      myString = 'Colgate 13'
      print(myString[0:7])  # Outputs Colgate
      print(myString[8:10])  # Outputs 13
      ```
    - What happens if you excluding the starting index: e.g., myString[:3]? --- slice starts at the beginning of the string: e.g., Col
    - What happens if you excluding the ending index: e.g., myString[6:]? --- slice ends at the end of the string: e.g., e 13
What is the output of the following programs?

a) `olympics = 'Pyeongchang'`
   ```python
   print(olympics[1])
   print(olympics[10])
   print(olympics[0] + olympics[8] + olympics[4])
   print(olympics[-1])
   print(olympics[-6])
   ```
   Output:
y
g
Pan
g
G

b) `sport = 'figure skating'`
   ```python
   print(sport[10:13])
   print(sport[:6])
   print(sport[8] + sport[11:])
   print(sport[-6:-3])
   ```
   Output:
tin
figure
king
Kat

c) `def pick_two(a, b, c):
   return a[b] + a[c]`
   ```python
   def remove(d, e, f):
      return d[:e] + d[f:]
   ```
   ```python
   def main():
      event = 'freestyle skiing'
      print(pick_two(event, -3, 4))
      print(remove(event, 4, -3))
      print(pick_two(remove(event, 0, 10), 2, 4))
      main()
   ```
   Output:
is
freeing
in

Is the following code valid? Why or why not? --- no, strings are immutable
```python
medal = 'gold'
course[3] = 'f'
```
String methods
- Can perform operations on string, similar to performing operations on a turtle
  - Example:
    ```python
    medal = 'silver'
    print(medal.upper())  # Outputs SILVER
    ```
- Common operations
  - `upper, lower` --- converts all characters to upper or lower case
  - `strip, lstrip, rstrip` --- removes leading/trailing whitespace (space, tab, newline)
  - `replace` --- replace all occurrences of one substring with another
  - `find, rfind` --- get index where substring is found
- What is the output of the following programs?
  a) `track = 'bobsled'
     print(track.replace('bob', 'mi'))
     print(track.replace('bob', ''))
     print(track.replace('b', 't'))
     print(track.find('sled'))
     print(track.find('b'))
     ```
     Output:
     misled
     sled
     totsled
     3
     0
   b) `sport = ' Ice hockey	'
     print('*' + sport.upper() + '*')
     print('*' + sport.strip() + '*')
     print('*' + sport.rstrip() + '*')
     print('*' + sport[:4].lower() + '*')
     ```
     Output:
     * ICE HOCKEY *
     *Ice hockey*
     * Ice hockey*
     * ice*
c) def emphasis(a):
    b = a.find(' ')
    c = a[:b].upper()
    return c + a[b:]

def devour(x, y):
    x = x.strip()
    return x.replace(y, '-' * len(y))

def main():
    sport = 'speed skating'
    print(emphasis(sport))
    print(devour(sport, 'ing'))
    print(devour(emphasis(sport), 's'))

main()
Output:
SPEED skating
speed skat---
SPEED -kating

String comparison
- Equality --- use double equals (==) to check if the strings are exactly the same
- Ordering --- use less than (<), greater than (>), etc. to determine whether one string comes before or after the other according to the lexicographical order
  - Lexicographic order is based on the numeric value a computer uses to represent a character
  - ord --- gets the numeric value for a character
  - chr --- gets the character for a numeric value
- Contains --- use in to determine whether one string contains another; use not in for the reverse
- Which of the following comparisons are True? (Hint: ord('A') returns 65, ord('Z') returns 90, and ord('a') returns 97.)
  a) 'alpha' == 'ALPHA'
     False
  b) 'Z' < 'A'
     False
  c) 'Z' < 'a'
     True
  d) 'bat' > 'baa'
     True
  e) 'moo' < 'Cow'
     False
  f) 'Moo' > 'MoO'
     True
  g) 'a' in 'apple'
     True
  h) 'na' in 'BaNaNa'
     False
i) 'cherry' in 'che'
   False
j) 'dragon' not in 'dragon fruit'
   False