MY470 Computer Programming

Working with Strings and Lists in Python

Week 2 Lab, MT 2017

Git

You can use your favourite editor by customizing Git setting. You should run those commands in console windows:

MacOS

```
git config --global core.editor "atom --wait"
```

Windows

```
git config --global core.editor notepad.exe
```

Alternatively

```
git config --global core.editor "'C:\Program Files (x86)\Notepad++\notepad++.exe'
 -multiInst -notabbar -nosession -noPlugin"
```

Variables

Variables associate objects (values) with a name. Objects have types (belong to classes).

Here are some rules and conventions for naming variables:

- Variables must begin with a letter (a - z, A - Z) or underscore (_)
- Variables can contain letters, underscore, and numbers

- Watch out for reserved words!

In [ ]:
```
# List of reserved words in Python: and, as, assert, break, class, continue, def,
 del, elif, else, except, exec,
# finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, ra
ise, return, try, while, with, yield

trial = 2
try = 5
```

Best Practice

- Use **ALLCAPS** for constants, like passwords or secret keys
- Use a consistent style, such as **mixedCaseName** or **underscore_name**

Resources

In addition to the Python resources online, you can query any object to get help on what methods are available

In [ ]:
```
dir(dict)
help(dict.popitem)
```

Strings

- Ordered sequences of characters
- Immutable

In [ ]:
```
x = 'my string'
x.capitalize()
print(x)
print(x[3])
print(x[1:-1])
print(x[::2])
```

In [53]:
```
# Exercise: Make three new strings from the first and last, second and second to l
ast, and
# third and third to last letters in the string below. Print the three strings.

palindrome = 'redder'
```

In [54]:
```
# Exercise: Make a new string that is the same as string1 but with the 8th and 22n
d characters missing.

string1 = 'I cancelled my travelling plans.'
```

String Methods

- `S.upper()`
- `S.lower()`
- `S.capitalize()`
- `S.find(S1)`
- `S.replace(S1, S2)`
- `S.strip(S1)`
- `S.split(S1)`
- `S.join(L)`

In [55]:
```
# Exercise: Remove the trailing white space in the string below, replace all doubl
e spaces with single space,
# and format to a sentence with proper punctuation. Print the resulting string.
string1 = '  this  is a very badly.  formatted string -  I would  like to make it
 cleaner\n'
```

Methods Can Be "Stringed"

However, be aware that this may reduce the clarity of your code.

It iss largely a question of style.

Except when you are working with large data — it is then also a question of memory.

In [56]:
```
# Exercise: Convert the string below to a list

s = "['apple', 'orange', 'pear', 'cherry']"
```

In [57]:
```
# Exercise: Reverse the strings below.

semordnilap1 = 'stressed'
semordnilap2 = 'drawer'
```

Lists

- Ordered sequence of values
- Mutable

In [ ]:
```
mylist = [1, 2, 3, 4]
mylist.append(5)
print(mylist)
```

List Methods

- `L.append(e)`
- `L.extend(L1)`
- `L.insert(i, e)`
- `L.remove(e)`
- `L.pop(i)`
- `L.sort()`
- `L.reverse()`

In [59]:
```
# Exercise: Use a list operation to create a list of ten elements, each of which i
s '*'
```

In [60]:
```
# Exercise: Assign each of the three elements in the list below to three variables
 a, b, c
ls = [['dogs', 'cows', 'rabbits', 'cats'], 'eat', {'meat', 'grass'}]
```

In [61]:
```
# Exercise: Replace the last element in ls1 with ls2
ls1 = [0, 0, 0, 1]
ls2 = [1, 1, 1]
```

In [62]:
```
# Exercise: Create a new list that contains only unique elements from list x

x = [1, 5, 4, 5, 6, 2, 3, 2, 9, 9, 9, 0, 2, 5, 7]
```

In [63]:
```
# Exercise: Print the elements that occur both in list a and list b

a = ['red', 'orange', 'brown', 'blue', 'purple', 'green']
b = ['blue', 'cyan', 'green', 'pink', 'red', 'yellow']
```

In [64]:
```
# Exercise: Print the second smallest and the second largest numbers in this list
 of unique numbers

x = [2, 5, 0.7, 0.2, 0.1, 6, 7, 3, 1, 0, 0.3]
```

In [65]:
```
# Exercise: Create a new list c that contains the elements of list a and b
# Watch out for aliasing - you need to avoid it here

a = [1, 2, 3, 4, 5]
b = ['a', 'b', 'c', 'd']
```

In [ ]:

```
# Extra: find the most efficient way
import time
n = 1000
size = 100000
print('Create a list of %d integers for %d times' % (size, n))

t = time.time()

for h in range(0, n):
    l1 = []
    for i in range(0, size):
        l1.append(99)

#print('l1 has ', len(l1), 'values')
print('Append %.2f millisecond' % ((time.time() - t) * 1000 / n))

t = time.time()

for h in range(0, n):
    l2 = []
    for i in range(0, size):
        l2.extend([99])

#print('l2 has ', len(l2), 'values')
print('Extend %.2f millisecond' % ((time.time() - t) * 1000 / n))

t = time.time()

for h in range(0, n):
    l3 = [0] * size
    for i in range(0, size):
        l3[i] = 99

#print('l3 has ', len(l3), 'values')
print('Init with zero %.2f millisecond' % ((time.time() - t) * 1000 / n))
```

```
t = time.time()

for h in range(0, n):
    l4 = [None] * size
    for i in range(0, size):
        l4[i] = 99

#print('l4 has ', len(l4), 'values')
print('Init with None %.2f millisecond' % ((time.time() - t) * 1000 / n))
```

Week 2 Assignment

- Practice string and list manipulations
- Practice working with data