Session 12 Lecture Notes

Announcements

1. Reading:
   (a) The Python Tutorial
      i. Section 3, *An Informal Introduction to Python*
      ii. Section 4, *More Control Flow Tools*
2. In-class lab exercise on Friday.
3. Second quiz on 2020-03-02, covers homework through Homework 6
4. Supplemental materials:
   (a) Think Python 2e by Allen Downey

Python sequence data types

A **sequence** data type is an **ordered** collection of objects.

Individual objects are accessed by an **index number**, starting with 0 for the first element.

```
first_element = my_sequence[0]      # get one object
shorter_sequence = my_sequence[1:3] # get a slice
spam = len(my_sequence)             # get number of objects
```

List

A `list` is a mutable sequence of arbitrary objects.

```
my_empty_list = []
my_foods = [ "spam", "eggs", "ham" ]
spiced_ham = my_foods[0]
my_foods[2] = "beans"
my_foods.append("spam") # more spam!
my_foods.append(4.95)   # For under five bucks!
print(repr(my_foods))
# prints ['spam', 'eggs', 'beans', 'spam', 4.95]
```

Strings

An `str` is an immutable sequence of **characters**

```
my_string = "Monty"
my_show = my_string + ' Python'
my_string_with_quotes = 'Knights who say "Ni!"'
my_index = my_string_with_quotes.find("say") # returns 12
end_of_line = "\n\r"
```

The `bytes` type is an immutable sequence of 8-bit unsigned integers.

```
three_nulls = b"\x00\x00\x00"
my_weapon = b'Holy Hand Grenade'
my_show_in_bytes = my_show.encode()  # str to bytes
my_weapon_str = my_weapon.decode()   # bytes to str
some_string.strip()      # remove leading/trailing whitespace
```

Tuples

A `tuple` is an immutable set of values

```
my_empty_tuple = ()
my_one_tuple = (8675309,)  # The comma is required
my_triple = ("spam", "eggs", 2.95)
entree, appetizer, price = my_triple
print((pin.value, voltage)) # for mu editor plotter
```

Conditional statements

The `if` statement includes optional `elif` and `else` clauses.

```
if predicate1:
  some statements
elif predicate2:
  some statements
elif predicate3:
  some statements
else:
  some statements
```

A **predicate** is an expression that can be evaluated to `True` or `False`.

Note that `==` and `!=` tests if two objects have the same **value**; `is` and `is not` test if two objects are actually the same object.

Complex predicates can be formed using the `not`, `and`, and `or` operators.

An empty sequence or a value of zero evaluates as `False`.

Loops

The `for` statement iterates over a sequence.

```
for my_value in my_sequence:     # for each sequence object
  print("My value is {0}".format(my_value))
```

```
for my_index, my_value in enumerate(my_sequence):
  print("Element {0} is {1}".format(my_index, my_value))
```

```
for my_int in range(my_range):   # for 0 to my_range-1
  print(my_int)
```

Note that using the `range` function allows you to avoid wordy loops like this:

```
my_int = 0
while my_int < my_range:
  print(my_int)
  my_int = my_int + 1
```

The `while` loop repeats until the predicate is false.

```
cans_of_spam = 100
while cans_of_spam > 0:
  print("We have spam!")
  cans_of_spam -= 1  # augmented assignment
```

The important work of an embedded process will be inside an infinite loop.

```
while True:          # infinite loop
  some statements
```