Lists

David H Smith IV

University of Illinois Urbana-Champaign

Tues, Thu 21 2021

Reminders

Reminders

Lists: General Review

Poll Question: List Function

What is the value of x?

```
1 x = list('abc')
```

A. 'abc'

B. ['abc']

C. ['a', 'b','c']

D. []

List Methods

Use `help(list.<method name>)` for information on a given method:

- `L.append(elem)` → Add element to the end of L.
- `L.extend(lst)` → Add all elements of lst to the end of L.
- `L.insert(index, elem)` → Insert element at index of L pushing other elements forward.
- `L.pop()` → Remove and return the element at the end of L.
- `L.pop(index)` → Remove and return the element at index of L.
- `L.remove(elem)` → Remove first occurrence of element from L.
- `L.sort(elem)` → Sort the elements of L.

Poll Question: List Functions

What is the value of `a` after this code is run?

```
1 a = [2, 4, 6, 8]
2 a.remove(4)
3 a.pop(2)
```

- A `[2, 4]`
- B `[6, 8]`
- C `[2, 6]`
- D `[2, 8]`

Poll Question: List Slicing/Indexing

What is the value of A after this code executes?

```
A = [1, 2, 3]
B = A
C = A[:]
B[1] = "pirate"
C[2] = "scurvy"
```

- A `[1, 2, 3]`
- B `[1, "pirate", 3]`
- C `[1, "pirate", "scurvy"]`
- D `["pirate", "scurvy"]`

Poll Question: More List Functions

Which will cause `x = [1, 2, 3, 4]`

```
1 x = [1, 2]
```

A. `x.append([3, 4])`

B. `x += [3, 4]`

C. `x.extend([3, 4])`

D. A and B

E. A and C

F. B and C

Poll Question: Loops

What will this output?

```
1 A = [5, 10, 15]
2 for i in range(len(A)):
3   A[i] = A[i] + 1
4 print("1: ", A, end=" ")
5
6 A = [5, 10, 15]
7 for e in A:
8   e = e + 1
9 print("2: ", A)
```

(A) `1: [5,10,15] 2: [5,10,15]`

(B) `1: [6,11,16] 2: [6,11,16]`

(C) `1: [5,10,15] 2: [6,11,16]`

(D) `1: [6,11,16] 2: [5,10,15]`

Poll Question: Removal Functions

We want a function that removes all spaces form strings. Which is correct?

```
1  def A(s):
2    for c in s:
3      if c == "":
4        c = ""
5    return s
6
7  def B(s):
8    new_s = ""
9    for c in s:
10     if c != " ":
11       new_s = new_s + c
12   return new_s
13
14 def C(s):
15   for i in range(len(s)):
16     if s[i] == " ":
17       s[i] = ""
18   return s
```

- A A
- B B
- C C
- D A and C
- E B and C

List Creation vs Modification

Creating vs Modifying Lists

Imagine lists `x` and `y`.

Creates new list:

A. `z = x.copy()`

B. `z = x[:]`

C. `z = x + y`

D. `z = sorted(x)`

E. `z = reversed(x)`

Modifies a list:

A. `x.sort()`

B. `x.append(num)`

C. `x.remove(num)`

D. `x.extend([num1, num2, ...])`

E. `x.pop(index)`

F. `x.insert(num)`

Nested Lists

Poll Question: Nested Lists

Which of the following is used to access 'e'?

```
my_list = [
    ['a', 'b', 'c'],
    ['d', 'e', 'f'],
    ['g', 'h', 'i']
  ]
x = my_list[1][2]
```

A. `my_list[2][2]`

B. `my_list[1][1]`

C. `my_list[1][2]`

D. None of the above

Poll Question: Nested Lists

What is the result of x?

```
my_list = [
    ['a', 'b', 'c'],
    ['d', 'e', 'f'],
    ['g', 'h', 'i']
  ]
x = my_list[1][2]
```

A. 'b'

B. 'd'

C. 'f'

D. 'h'

Poll Question: Nested Lists

What is the resulting value of z after this code runs?

```
my_list = [
    ['a', 'b', 'c'],
    ['d', 'e', 'f'],
    ['g', 'h', 'i']
  ]
z = ""
for y in my_list:
  for x in y:
    total += x
```

A. 'abcdefghi'

B. ['abc','def','ghi']

C. ['[abc]','[def]','[ghi]']

D. '[abcdefghi]'

numpy

numpy.array vs lists

1. Shape in numpy:
   - Imagine we have a numpy array `a`.
   - `a.shape[0]` height of the array (y dimesion).
   - `a.shape[1]` width of the array (x dimension).

numpy.array vs lists

1. Shape in numpy:
   - Imagine we have a numpy array `a`.
   - `a.shape[0]` height of the array (y dimesion).
   - `a.shape[1]` width of the array (x dimension).
2. To index into a 2d array: `regular_python_list[y][x]` $\longrightarrow$ `np_array[y, x]`

numpy.array vs lists

1. Shape in numpy:
   - Imagine we have a numpy array `a`.
   - `a.shape[0]` height of the array (y dimesion).
   - `a.shape[1]` width of the array (x dimension).
2. To index into a 2d array: `regular_python_list[y][x]` $\longrightarrow$ `np_array[y, x]`
3. To get a chunk of an numpy array:

   ```
   np_array[y_start: y_end, x_start: x_end]
   ```

numpy.array vs lists

1. Shape in numpy:
   - Imagine we have a numpy array `a`.
   - `a.shape[0]` height of the array (y dimesion).
   - `a.shape[1]` width of the array (x dimension).
2. To index into a 2d array: `regular_python_list[y][x]` $\rightarrow$ `np_array[y, x]`
3. To get a chunk of an numpy array:

   ```
   np_array[y_start: y_end, x_start: x_end]
   ```
4. `numpy.zeros((y, x))` $\rightarrow$Produces a 2d numpy array of x by y dimensions

numpy.array vs lists

1. Shape in numpy:
   - Imagine we have a numpy array `a`.
   - `a.shape[0]` height of the array (y dimesion).
   - `a.shape[1]` width of the array (x dimension).
2. To index into a 2d array: `regular_python_list[y][x]` $\rightarrow$ `np_array[y, x]`
3. To get a chunk of an numpy array:
   `np_array[y_start: y_end, x_start: x_end]`
4. `numpy.zeros((y, x))` $\rightarrow$Produces a 2d numpy array of x by y dimensions
5. Other than that, they behave very similar to regular python lists but come with a ton of helpful functions (See `help(numpy.array)`)!