

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()` → Sort the elements of L.

Poll Question: List Slicing/Indexing

What is the value of A after this code executes?

```
1 A = [1, 2, 3]
2 B = A
3 C = A[:]
4 B[1] = "pirate"
5 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)
```

- Ⓐ 1: [5,10,15] 2: [5,10,15]
- Ⓑ 1: [6,11,16] 2: [6,11,16]
- Ⓒ 1: [5,10,15] 2: [6,11,16]
- Ⓓ 1: [6,11,16] 2: [5,10,15]

Poll Question: Removal Functions

We want a function that removes all spaces from a string. 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
- ☐ B
- ☐ C
- ☐ A and C
- ☐ 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'?

```
1 my_list = [  
2     ['a', 'b', 'c'],  
3     ['d', 'e', 'f'],  
4     ['g', 'h', 'i']  
5 ]
```

- ☐ 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?

```
1 my_list = [  
2     ['a', 'b', 'c'],  
3     ['d', 'e', 'f'],  
4     ['g', 'h', 'i']  
5 ]  
6 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?

```
1 my_list = [  
2     ['a', 'b', 'c'],  
3     ['d', 'e', 'f'],  
4     ['g', 'h', 'i']  
5 ]  
6 z = ""  
7 for y in my_list:  
8     for x in y:  
9         z += 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] → 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] → 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 dimension).
 - `a.shape[1]` width of the array (x dimension).
- ② To index into a 2d array: `regular_python_list[y][x] → 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))` → Produces a 2d numpy array of x by y dimensions

numpy.array vs lists

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