

Creative Computing I

Lists

Warmup

- When do I need to put `from random import *` at the top of my programs?
- What are the different random functions
- Why are lists useful?
- How do I reverse a list?

Tutorial

- Did that format help?
- I plan to put more reading materials up
- Activities: unless I explicitly ask to see them (like tutorial), they're for your practice
- I do grade on-taskness

Why lists?

- Group related values
- Access multiple values through one variable
- Avoid tedious if/elif sequences
- Note: I may use the word 'array' since that's what they're called in C/C++/Java

Creating a list

- Put a comma-separated list of expressions in square brackets
- For an empty list, put nothing in brackets

```
my_list1 = []
```

```
my_list2 = [expression, ...]
```

```
seattle_temps = [54, 51, 47, 48, 53, 54, 46]
```

index	0	1	2	3	4	5	6
value	54	51	47	48	53	54	46

Accessing list items

- Uses bracket notation []
- Careful, lists use 0-based indexing!
- Each item keeps original type

```
print("Sunday's temp: " + str(seattle_temps[0]))  
tues_temp = seattle_temps[2]
```

index	0	1	2	3	4	5	6
value	54	51	47	48	53	54	46

Printing a list

- Can print a list directly but it's always comma-separated with brackets

```
print(seattle_temps)
```

```
[54, 51, 47, 48, 53, 54, 46]
```

Iterating over lists

- For loops are made to use with lists
- We've used them with `range`, but `range` just creates a list!!

```
for <item> in <list>:           54
    print(<item>)                51
                                47

for temp in seattle_temps:      48
    print temp                   53
                                54
                                46
```

index	0	1	2	3	4	5	6
value	54	51	47	48	53	54	46

Range function

- The `range` function can give us a list
- Takes up to three parameters: a start, an end and a step
- Gives back a list of integers

```
my_list3 = range(4, 8, 2)
```

is equivalent to

```
my_list3 = [4, 6]
```

Functions on lists

- **Built-in Python functions that take lists as parameters**
- `len(<list>)` - number of items in list
- `max(<list>)` - biggest item in list
- `min(<list>)` - smallest item in list
- `sum(<list>)` - sum of all items in list

```
days = len(seattle_temps)
max_temp = max(seattle_temps)
print("Over the next " + str(days) +
      " days, the maximum temperature will be: " + str(max_temp))
```

Average

- Write a function `average` that takes a list as a parameter and prints out its average value
- `average(seattle_temps)` should print 50

Objects

- Lists are objects
- Powerful, modern programming paradigm!!
- Group state (variables) and behavior (methods)

`seattle_temps` →

index	0	1	2	3	4	5	6
value	54	51	47	48	53	54	46

`sort()` `pop()`
`push()` `reverse()`

Methods

- Like functions that we can call on the list
- We use 'dot notation' - `<var>.<method>()`
- These methods often modify the list

index	0	1	2	3	4	5	6
value	54	51	47	48	53	54	46

`seattle_temps.reverse()`

index	0	1	2	3	4	5	6
value	46	54	53	48	47	51	54

`seattle_temps.sort()`

index	0	1	2	3	4	5	6
value	46	47	48	51	53	54	54

`seattle_temps.pop()`

index	0	1	2	3	4	5
value	46	47	48	51	53	54

[All list methods: http://docs.python.org/tutorial/datastructures.html#more-on-lists](http://docs.python.org/tutorial/datastructures.html#more-on-lists)

Questions

- How do I prepend an item? (add it to the front)
- How do I print a list on the same line as a string?