Lists and loops

Presenter: Steve Baskauf
steve.baskauf@vanderbilt.edu

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

CodeGraf landing page

- vanderbi.lt/codegraf

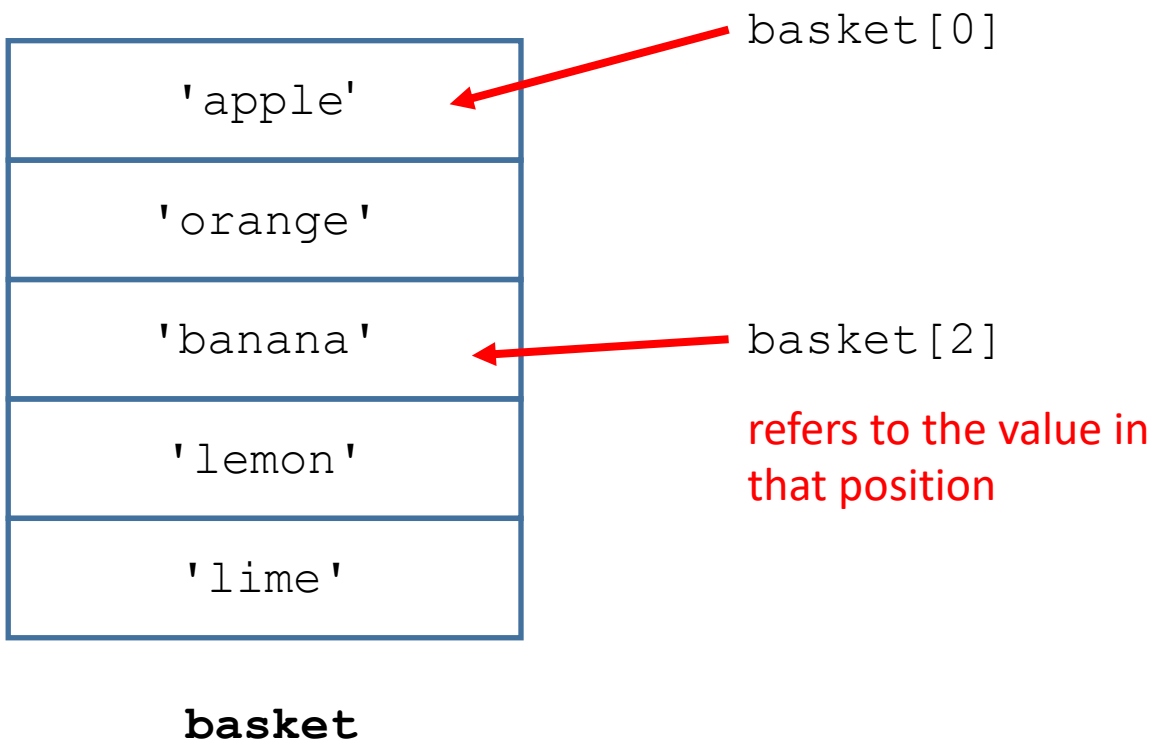
List objects

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

List objects

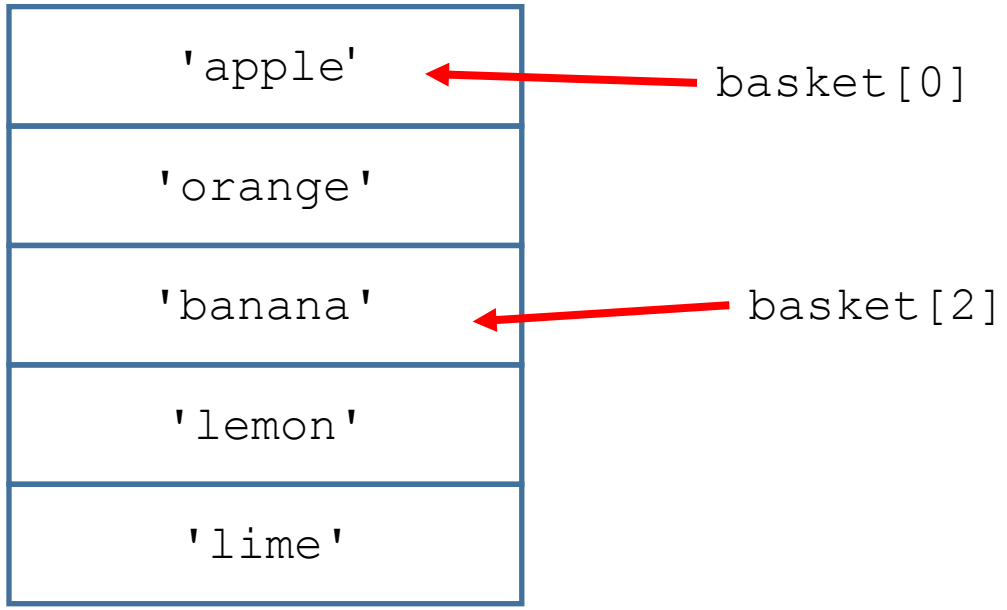
- A list object is a one-dimensional data structure
- Lists can hold any other kind of object.
- The items in a list are referred to by an index number (0-based)



Instantiating a list

- A list can be constructed directly by listing its contents.
- The type of the list is different from the type of items the list contains.
- List items don't have to all be of the same type (but often are).

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
```



Finding the length of a list

- The **`len()`** function will return the number of items in a list
- Example:

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
print(len(basket))
```

- Item indices range from 0 to 4
- Length is 5 (the actual count)
- In many ways, a string is like a list of characters; **`len()`** works for it

Other ways to make a list

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

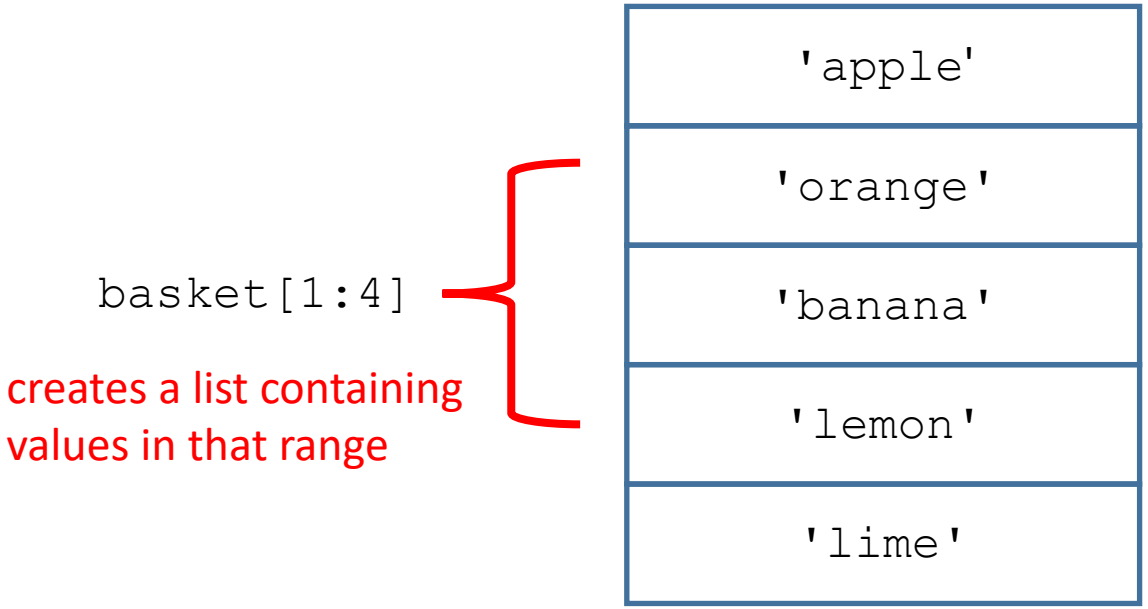
Jean & Alexander Heard LIBRARIES

Output of functions or methods

- The output of a function or method may be a list:
  - **`os.listdir()`** function
  - **`random.sample()`** function
  - **`.split()`** string method

Slicing a list

- A range is given instead of a single index
- Start of range is zero-based.
- End of range is one less than ending index.
- Slicing generates another list



Aside: slicing a string

- Since a string is like a list of characters, we can slice it in the same way
- Example:

```
a_word = 'Mississippi'
word_piece = a_word[1:4]
```

- Range is from 1 to 4
- Slice goes from letters 1 to 3 (start counting with 0)
- Answer: `'iss'`

Useful things to do with lists

- Randomize a list
  - **`random.shuffle()`** function
- Sort a list
  - **`.sort()`** list method
- Pick an item from a list
  - **`random.choice()`** function

Changing a list

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

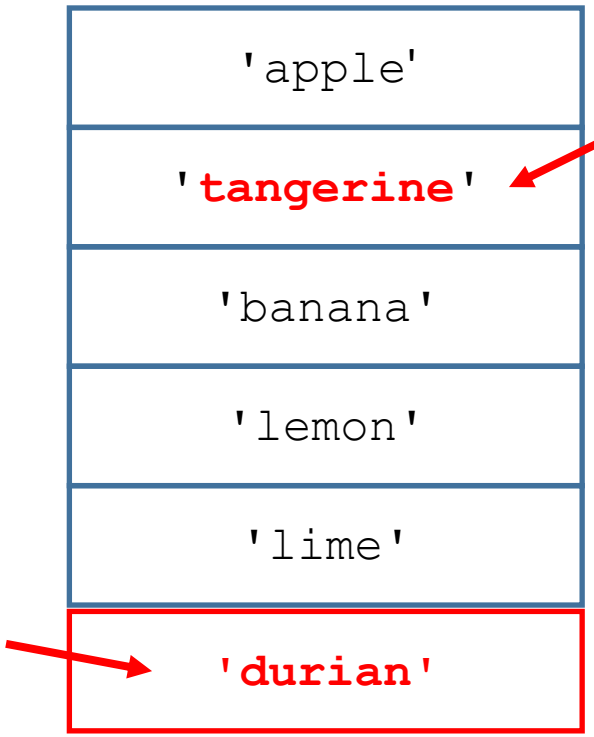
Jean & Alexander Heard LIBRARIES

Editing lists

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
```

```
basket[1] = 'tangerine'
```

We can assign a new value to any list item.




```
basket.append('durian')
```

The `.append()` method does not return a value – it changes the list.

More commands for editing lists

- An **empty list** can be created using

```
basket = []
```

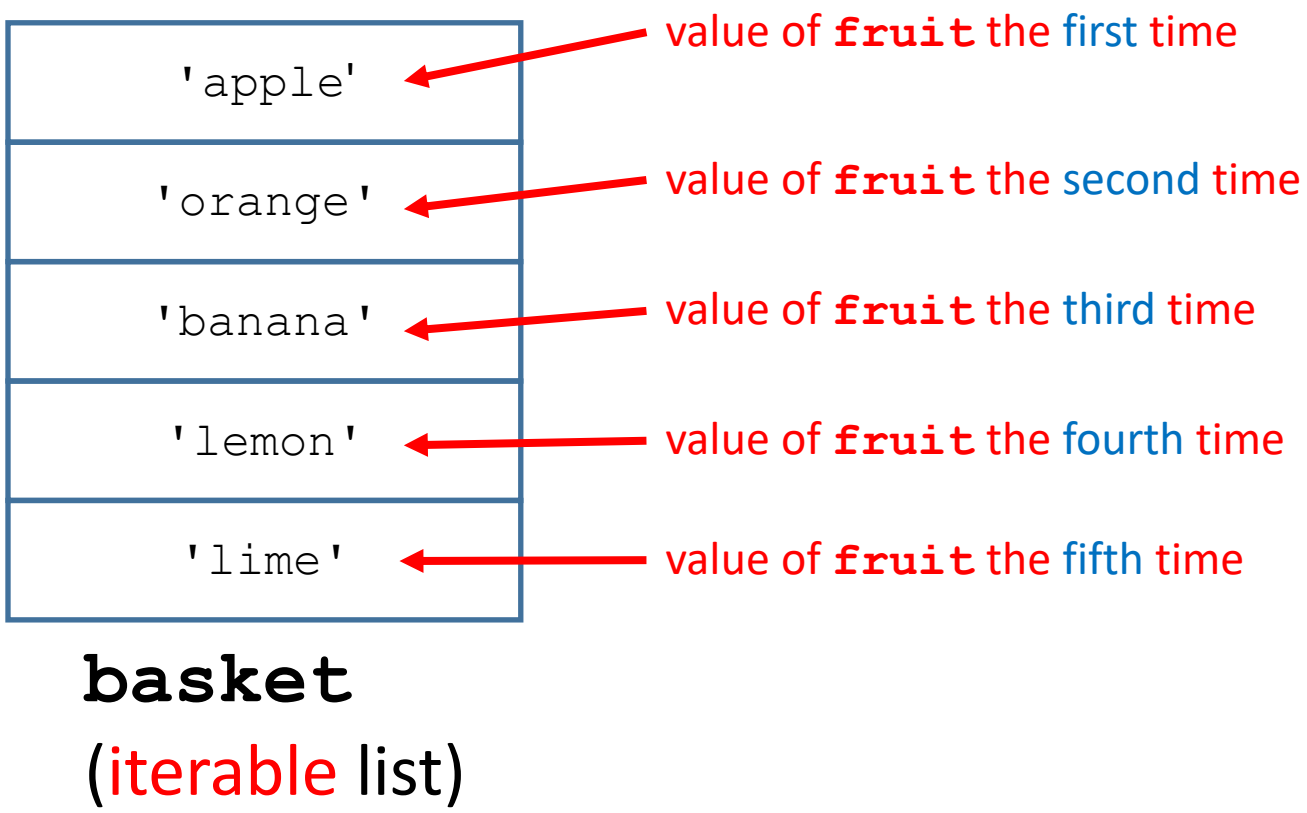
- `.remove()` can be used to remove a **particular value** from the list.
- `del basket[3]` can be used to remove an **item by position**

- The **+** operator appends the items in the second list to the end of the first list.

`for` loop

Iterating with **for**

```
for fruit in basket:
    do this indented code block once for each fruit
then do this code block
```



Example

notice this colon

```
basket = ['apple', 'orange', 'banana', 'lemon', 'lime']
for fruit in basket:
    print('I ate one ' + fruit)
print("I'm full now!")
```

- The indented code block can have more than one line.
- The upcoming code block is signaled by a colon (:)
- Strings are iterable by character.
- **`for`** is useful when there are a definite number of loops

`range()` as an iterable

- The range iterates from the first number to one step less than the second number:
  - `range(1, 11)` iterates from 1 to 10
- A step is optional:
  - `range(2, 10, 2)` iterates by twos from 2 to 8
- The step can be negative:
  - `range(10, 0, -1)` iterates from 10 to 1

Using the value of the range

```
for number in range(1, 11):
    the_square = number**2
    the_area = the_square * 3.14159
    print(number, '\t', the_area)
print("Those are the areas of all the circles!")
```

- The value of the iterated variable can be used anywhere in the indented code block.

Examples

- It's very common to use the length of a list as the end of a range (see last example).
  - Using the length of the list iterates through the whole list because counting is zero-based.

`while` loop

Looping with **while**

```
power = 0
exponent = 0
print('exponent\tpower')
while power < 100:
    power = 2**exponent
    exponent += 1
    print(exponent, '\t', power)
print("Those are the powers of two.")
```

- **while** loops are useful for an indefinite number of loops
- The test value must have an initial value.
- The test value must be able to meet the condition.
- The test is not made again until the loop end

Stringing together methods

DISC DIGITAL SCHOLARSHIP AND COMMUNICATIONS

Jean & Alexander Heard LIBRARIES

Applying methods sequentially

- Recall that functions can be nested inside functions.
- A method can be added onto a method.
- The output of the first method must be the correct type for the second method.
- Example:

```
from datetime import date
this_day = date.today().weekday()
```

Example using strings, lists, and numbers

- Stringing together methods makes compact code
- It also makes less readable code.
- Similar problem to nesting many functions:

```
sqrt(int(input('How many? ')))
```

Remote Support for Teaching and Research Needs

- Access to digital collections 24/7
- Skype consultations with your subject librarian
- Ask a Librarian: an easy way to submit a question via email
- Live chat available from the Library home page

Jean & Alexander Heard
LIBRARIES

NEED HELP? ASK A LIBRARIAN!

https://www.library.vanderbilt.edu/ask-librarian.php

Jean & Alexander Heard
LIBRARIES