COMP1021
Introduction to Computer Science

Lists and Tuples

David Rossiter and Gibson Lam

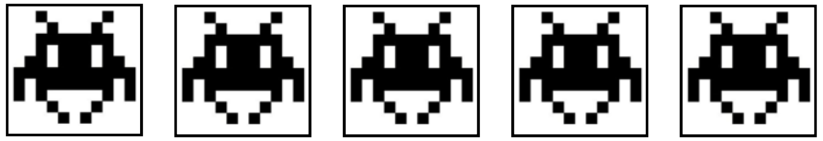
Outcomes

- After completing this presentation, you are expected to be able to:
  1. Manage a collection of things using a list or a tuple in Python
  2. Explain the major difference between lists and tuples
  3. Create a two dimensional structure using lists or tuples

Storing a Collection of Data

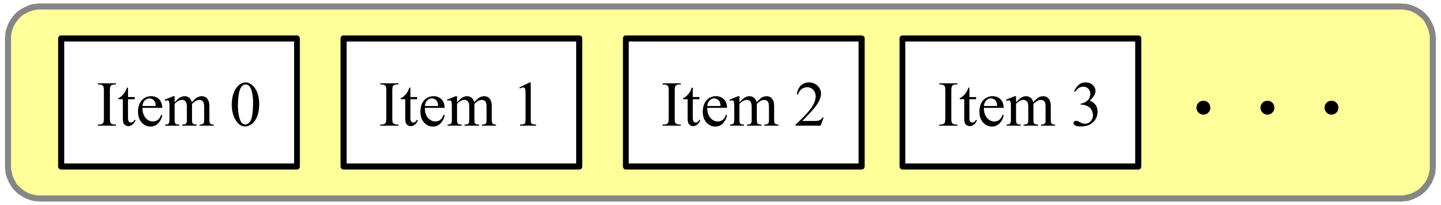
- You have seen the use of variables to store simple data such as a number or a string (a piece of text)
- Instead of a single piece of data it will be useful if you can store a collection of data in a variable inside a program
- For example, in a shooting game, if you can store a collection of monsters in one place, you will be able to manage them a lot easier than having them stored separately

`monsters =`

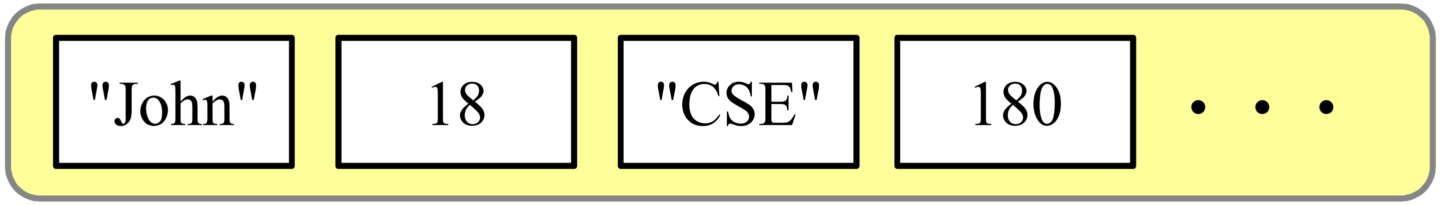


Lists and Tuples

- Lists and tuples are two different ways to store a collection of items in Python



- Both of them can be used to store many items, which can be of any types, at the same time
- For example, you can store a collection of numbers and text in a list or a tuple



Using a List

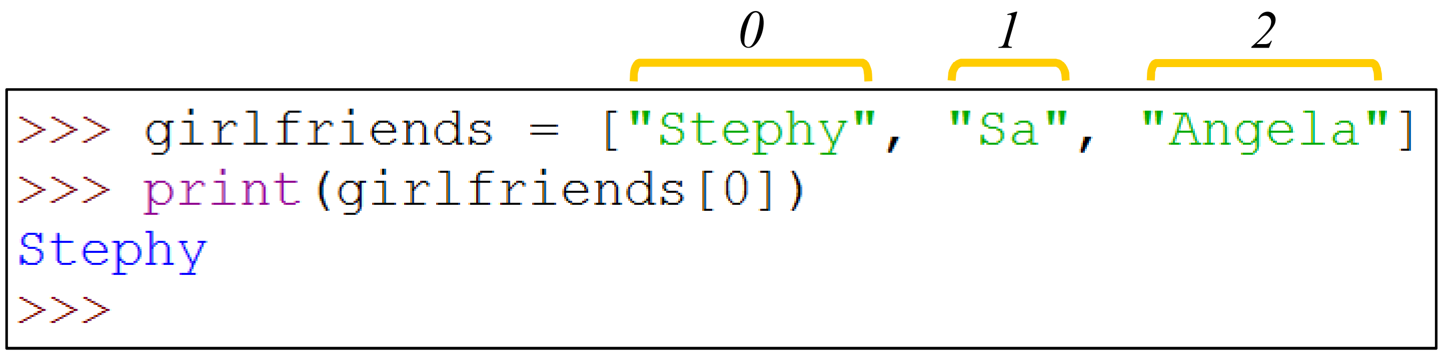
- To create a list in Python you use a pair of square brackets to enclose the items you need, for example:

```
girlfriends = ["Stephy", "Sa", "Angela"]
```

- To get back a particular item from the list, you need to say which item you want by giving an *index*
- For example, to print the first item from the above list, you can use this code:

```
print(girlfriends[0])
```

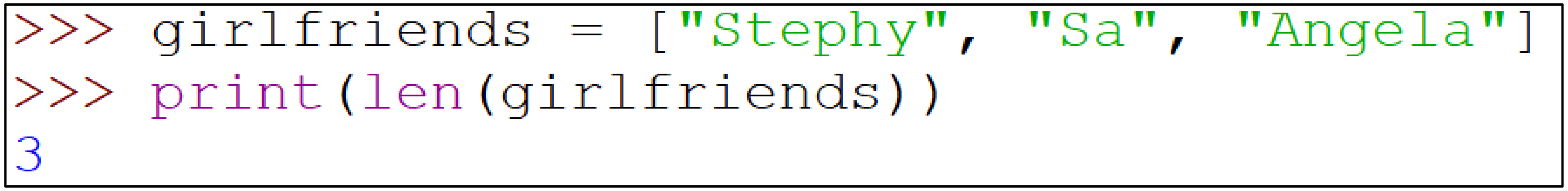
Important! The first item has an index of 0, not 1!



```
>>> girlfriends = ["Stephy", "Sa", "Angela"]
>>> print(girlfriends[0])
Stephy
>>>
```

The Length of a List/ Tuple

- You can use `len( name )` to tell you how many things are in a list/ tuple
- For example:



```
>>> girlfriends = ["Stephy", "Sa", "Angela"]
>>> print(len(girlfriends))
3
```

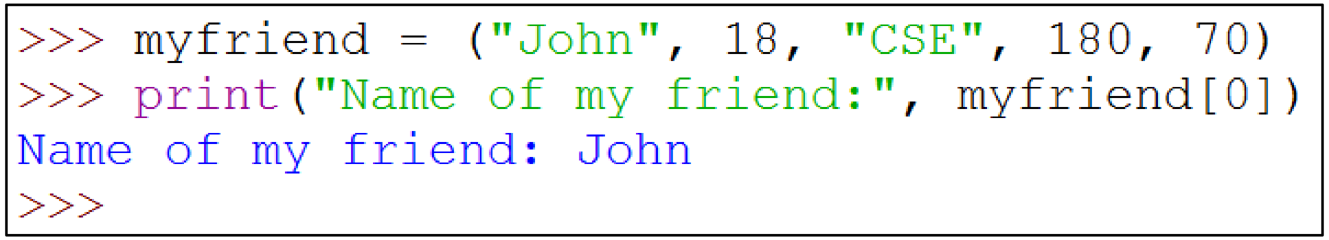
Using a Tuple

- To create a tuple you use a pair of parentheses (not square brackets!) for the items, like this:

```
myfriend = ("John", 18, "CSE", 180, 70)
```

- Again, to get a particular item from a tuple, you use an appropriate index, for example:

```
print("Name of my friend:", myfriend[0])
```



```
>>> myfriend = ("John", 18, "CSE", 180, 70)
>>> print("Name of my friend:", myfriend[0])
Name of my friend: John
>>>
```

You still use [ ] *when getting an item from a tuple*

Lists vs Tuples

- You may wonder what the difference between lists and tuples is (apart from the brackets/parentheses)
- One major difference is that:
  - After making a list you can modify the list later (adding, changing or deleting an item)
  - After making a tuple you **cannot** modify the tuple in any way, i.e. a tuple is read-only
- Using the language of computing, we say lists are *mutable* (can be changed) and tuples are *immutable* (cannot be changed)

Changing an Item in a List

- You can change any item in a list, for example:

```
>>> girlfriends = ["Stephy", "Sa", "Angela"]
>>> girlfriends[2] = "GEM"
>>> print(girlfriends)
['Stephy', 'Sa', 'GEM']
>>>
```

- However, you cannot do that for a tuple, like this:

```
>>> myfriend = ("John", 18, "CSE", 180, 70)
>>> myfriend[2] = "ECE"
Traceback (most recent call last):
  File "<pyshell#16>", line 1, in <module>
    myfriend[2] = "ECE"
TypeError: 'tuple' object does not support item assignment
>>> print(myfriend)
('John', 18, 'CSE', 180, 70)
>>>
```

Insert, Remove and Append a List

- You can insert/remove/append items at any time, for example:

'append' means 'put it at the end'

```
>>> comment = ["Dave", "is", "human"]
>>> print(comment)
['Dave', 'is', 'human']
>>> comment.insert(2, "not")
>>> print(comment)
['Dave', 'is', 'not', 'human']
>>> comment.remove("human")
>>> print(comment)
['Dave', 'is', 'not']
>>> comment.append("evil!")
>>> print(comment)
['Dave', 'is', 'not', 'evil!']
```

Insert before item 2 (the third item)

If there was more than one 'human' in the list, this would only remove the first one

Other Things You Can Do with Lists

- sort – sorts the list in increasing number/letter order

```
>>> words = ["cat", "dog", "apple", "bat"]
>>> words.sort()
>>> print(words)
['apple', 'bat', 'cat', 'dog']
```

- reverse – reverses the content of the list

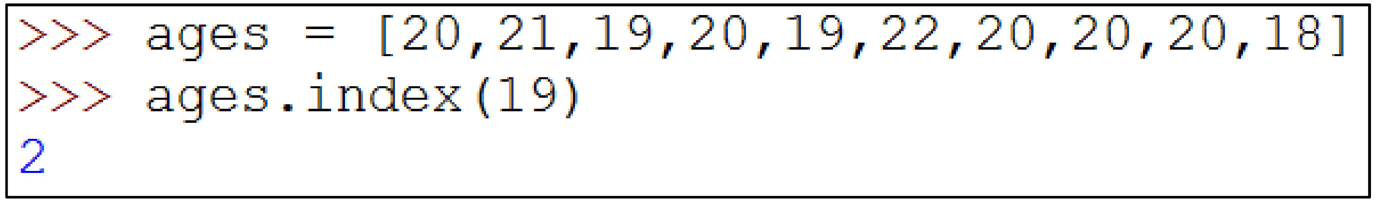
```
>>> words = ["cat", "dog", "apple", "bat"]
>>> words.reverse()
>>> print(words)
['bat', 'apple', 'dog', 'cat']
```

- count – counts how many times something appears

```
>>> ages = [20,21,19,20,19,22,20,20,20,18]
>>> ages.count(20)
5
```

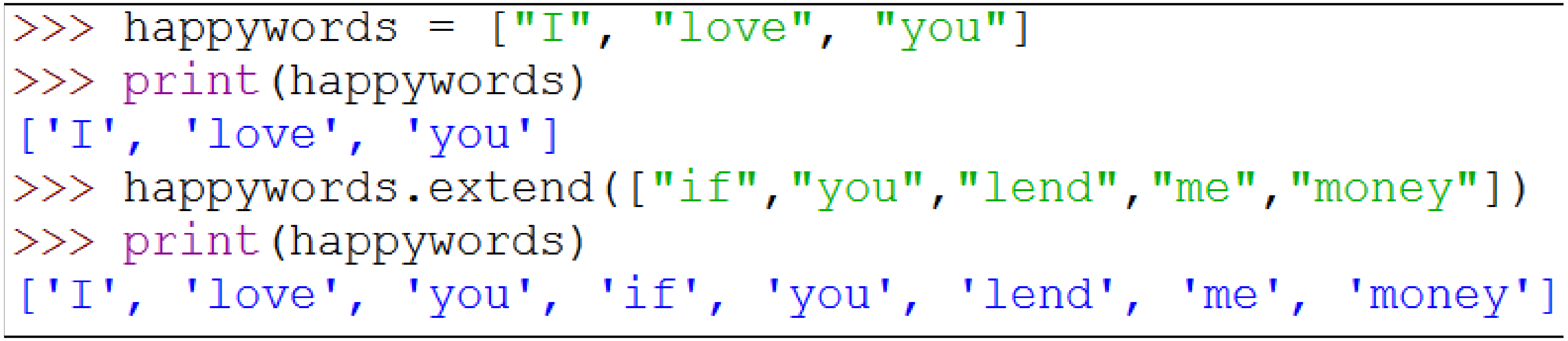
Other Things You Can Do with Lists

- index – finds the first occurrence of something



```
>>> ages = [20,21,19,20,19,22,20,20,20,18]
>>> ages.index(19)
2
```

- extend – appends another list to this list



```
>>> happywords = ["I", "love", "you"]
>>> print(happywords)
['I', 'love', 'you']
>>> happywords.extend(["if","you","lend","me","money"])
>>> print(happywords)
['I', 'love', 'you', 'if', 'you', 'lend', 'me', 'money']
```

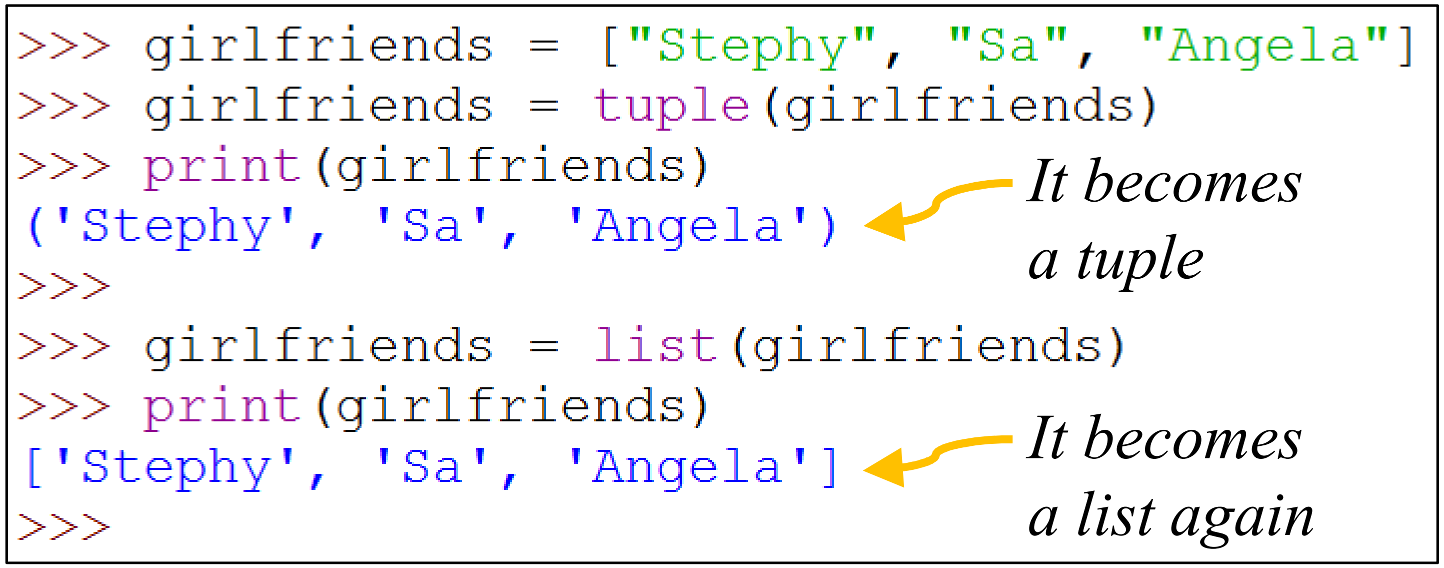
- There is also ‘pop’, which we will look at in another presentation

When to Use Lists or Tuples

- Up to this point, you may simply stick to using lists because they are ‘more powerful’ than tuples
- However, there are situations when you really only need to use tuples because you don’t need to change the data in any way
- In addition, sometimes Python does not allow the use of lists, e.g. sometimes in a dictionary (discussed in another presentation)

Converting Between Lists and Tuples

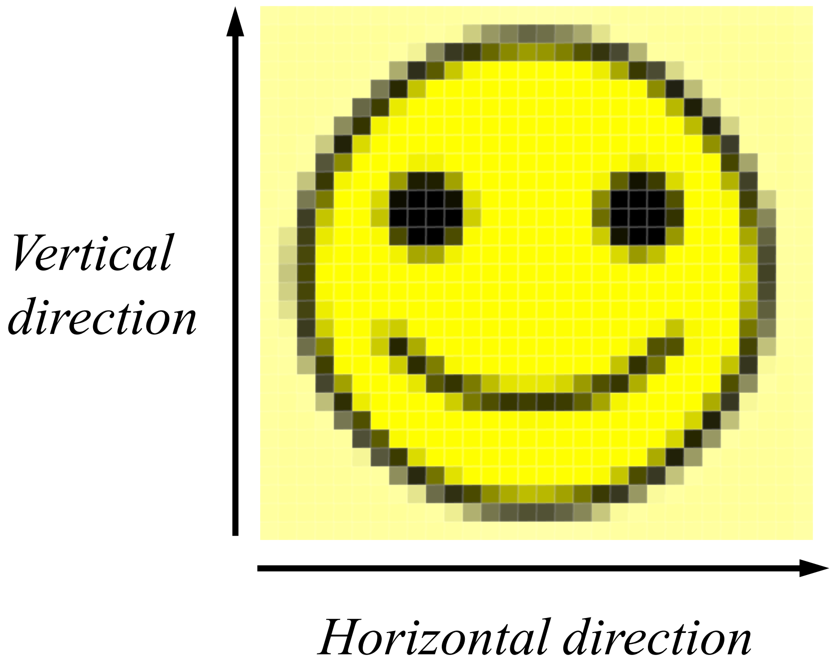
- Sometimes you may want to change a tuple to a list or vice versa
- To do that you can use the list command or tuple command
- Here is an example:



```
>>> girlfriends = ["Stephy", "Sa", "Angela"]
>>> girlfriends = tuple(girlfriends)
>>> print(girlfriends)
('Stephy', 'Sa', 'Angela')
>>>
>>> girlfriends = list(girlfriends)
>>> print(girlfriends)
['Stephy', 'Sa', 'Angela']
>>>
```

Two Dimensional Structures

- In some cases a one dimensional (1D) structure (things that are arranged in one direction) is not enough
- For example, a digital camera image is a 2D structure:



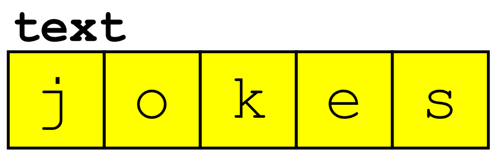
Two Dimensional Structures

- A Python list or tuple is a 1D data structure
- What if you want to use a 2D structure?
  - Then you need to use lots of lists/tuples inside another list/tuple
- For example, you can create a list of lists
  - The outer list is one of the dimensions, and the inner lists are the other dimension

Examples of a 1D List and a 2D List

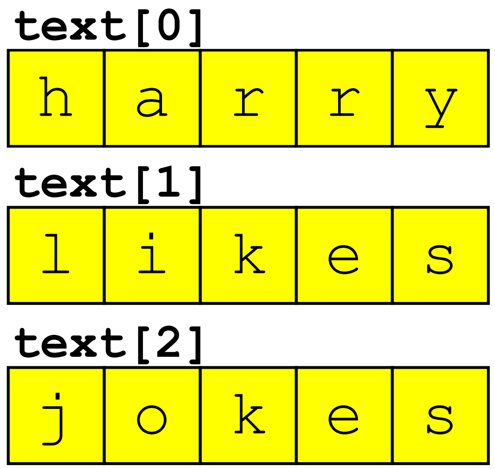
- 1D example:

```
text = ["j","o","k","e","s"]
letter_o = text[1] # the "o" in second col
```



- 2D example:

```
text =
 [["h","a","r","r","y"],
  ["l","i","k","e","s"],
  ["j","o","k","e","s"]]
letter_o = text[2][1]
# the "o" in the third row second column
```



Be careful! The general idea is `[row][col]`, not `[col][row]`!