COMP1021
Introduction to Computer Science

Lists, Tuples and Strings

David Rossiter and Gibson Lam

Outcomes

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

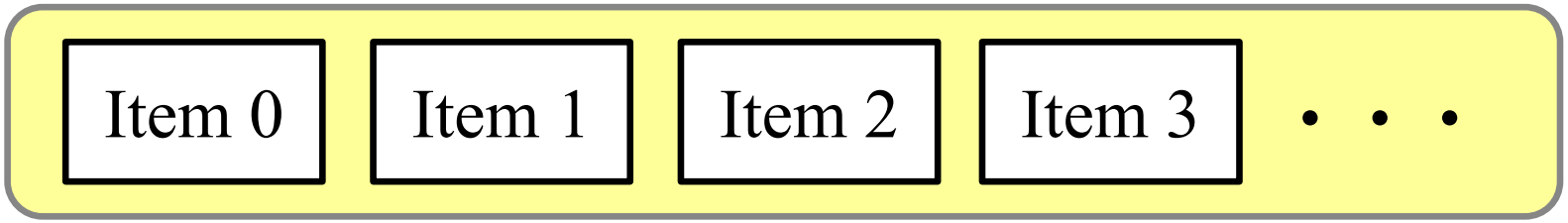
Storing a Collection of Data

- 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

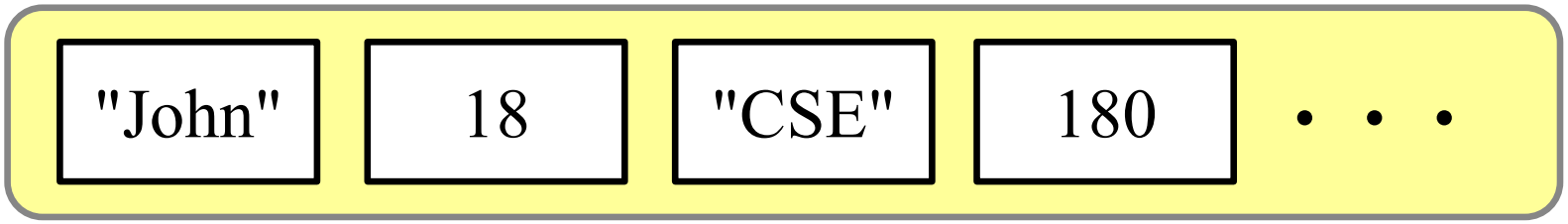
`monsters =` 

Lists

- Lists are a way to store a collection of items



- They can be used to store many items together
- The items can be of any type
- For example, you can store a collection of numbers and text together in a list:



Using a List

- To create a list in Python you use a pair of square brackets, for example:

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

0 *1* *2*

```
girlfriends = ["Gwen", "Lois", "Mary"]
```

- To access a particular item you need to use the *index*
- For example, to print the first item:

gf.py - //VDIDRIVE/MYHOME/rossiter/Documents/gf.py (3.10.4)

File Edit Format Run Options Window Help

```
girlfriends = ["Gwen", "Lois", "Mary"]
print(girlfriends[0])
```

```
========== RESTART:
Gwen
>>>
```

The Length of a List

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

gf2.py - //VDIDRIVE/MYHOME/rossiter/Documents/gf2.py (3.10.4)

File Edit Format Run Options Window Help

```
girlfriends = ["Gwen", "Lois", "Mary"]
print(len(girlfriends))
```

```
========== RESTART:
3
>>>
```

Going Through Everything in a List

- There's 2 ways to go through everything in a list
- The first way is shown here:

gf3.py - //VDIDRIVE/MYHOME/rossiter/Documents/gf3.py (3.10.4)

File Edit Format Run Options Window Help

```
girlfriends = ["Gwen", "Lois", "Mary"]
for thisgirlfriend in girlfriends:
    print(thisgirlfriend)
```

```
============= RESTART:
Gwen
Lois
Mary
>>>
```

Going Through Everything in a List

- The second way is by using *range* to generate the the index numbers, then using the index numbers:

gf4.py - //VDIDRIVE/MYHOME/rossiter/Documents/gf4.py (3.10.4)

File Edit Format Run Options Window Help

```
girlfriends = ["Gwen", "Lois", "Mary"]
for index in range(len(girlfriends)):
    print(girlfriends[index])
```

```
============= RESTART:
Gwen
Lois
Mary
>>>
```

- The first way is simpler than the second way, but sometimes you need to use the second way

Changing an Item in a List

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

manygf.py - //VDIDRIVE/MYHOME/rossiter/Documents/manygf.py (3.10.4)

File Edit Format Run Options Window Help

```
girlfriends = ["Gwen", "Lois", "Mary"]
girlfriends[2] = "Pepper"
print(girlfriends)
girlfriends[0] = "Emma"
print(girlfriends)
```

```
=============== RESTART: //VD
['Gwen', 'Lois', 'Pepper']
['Emma', 'Lois', 'Pepper']
>>>
```

You can use either " … " or ' … ' for text – see later

Insert, Remove and Append a List

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

'append' means 'put it at the end'

apple.py - //VDIDRIVE/MYHOME/rossiter/Documents/apple.py (3.10.4)

File Edit Format Run Options Window Help

```
words = ["An", "apple", "is", "tasty"]
print(words)
words.insert(3, "not")
print(words)
words.remove("apple")
words.insert(1, "ant")
words.append("probably!")
print(words)
```

Insert before item 3

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

```
============== RESTART: //VDIDRIVE/MYHOME/rossite
['An', 'apple', 'is', 'tasty']
['An', 'apple', 'is', 'not', 'tasty']
['An', 'ant', 'is', 'not', 'tasty', 'probably!']
```

Other Things You Can Do with Lists

- reverse – reverses the content of the list

reversedemo.py - //VDIDRIVE/MYHOME/rossiter/Documents/reversedemo.py (3.10.4)

File Edit Format Run Options Window Help

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

```
========== RESTART: //VDIDRIVE/M
['bat', 'apple', 'dog', 'cat']
>>>
```

Other Things You Can Do with Lists

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

sortdemo.py - //VDIDRIVE/MYHOME/rossiter/Documents/sortdemo.py (3.10.4)

File Edit Format Run Options Window Help

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

```
========== RESTART: //VDIDRIVE/M
['apple', 'bat', 'cat', 'dog']
>>>
```

Other Things You Can Do with Lists

- count – counts how many times something appears

countdemo.py - //VDIDRIVE/MYHOME/rossiter/Documents/countdemo.py (3.10.4)

File Edit Format Run Options Window Help

```
ages = [20,21,19,20,19,20,20,20,18]
print(ages.count(20))
```

```
>>> ======
5
```

Other Things You Can Do with Lists

- index – finds the index of the first occurrence of something

ages.py - //VDIDRIVE/MYHOME/rossiter/Documents/ages.py (3.10.4)

File Edit Format Run Options Window Help

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

```
>>> ========
2
```

Other Things You Can Do with Lists

- extend – appends another list to this list

nicewords.py - //VDIDRIVE/MYHOME/rossiter/Documents/nicewords.py (3.10.4)

File Edit Format Run Options Window Help

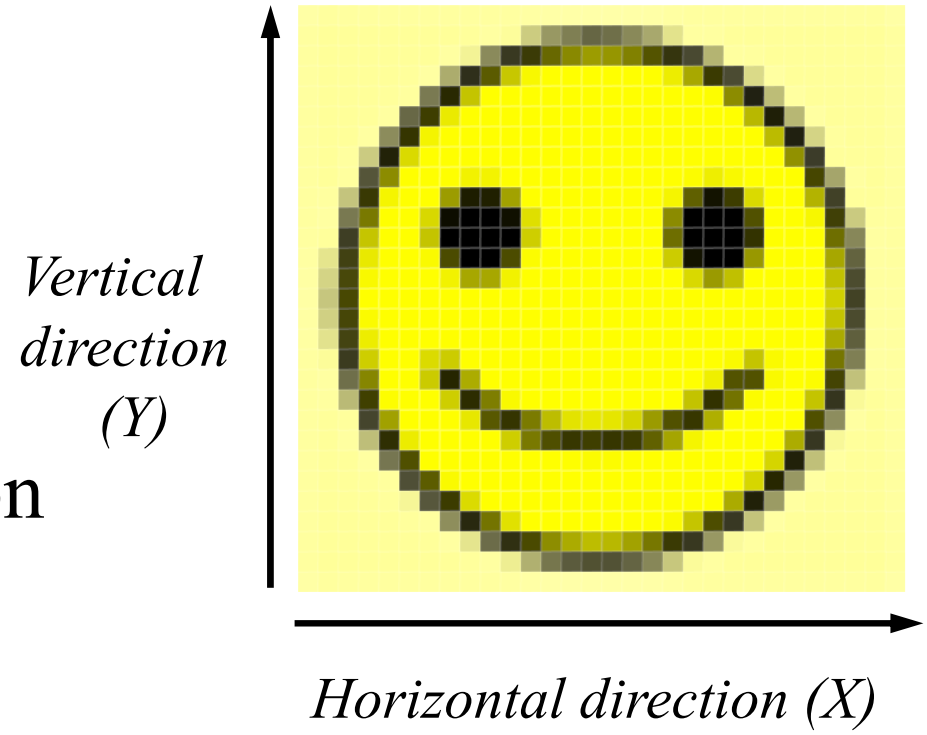
```
happywords = ["I", "love", "you"]
print(happywords)
happywords.extend( ["if","you","give","me","money"] )
print(happywords)
```

```
============ RESTART: //VDIDRIVE/MYHOME/rossiter/Documents
['I', 'love', 'you']
['I', 'love', 'you', 'if', 'you', 'give', 'me', 'money']
>>>
```

- There are other things you can do with lists, we may look at those later in the course if we have time

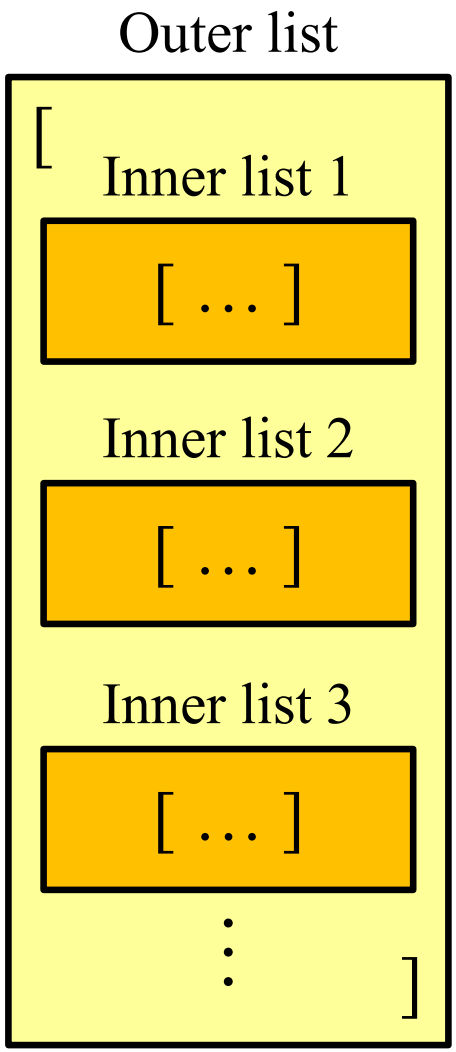
Two Dimensional Structures

- 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
  - You need both the X location and Y location of a point to read its colour in an image



Two Dimensional Structures

- A Python list is a 1D data structure
- What if you want to use a 2D structure?
  - Then you need to use lots of lists inside another list
- We call this 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:

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

things

j	o	k	e	s

- 2D example:

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

things[0]

h	a	r	r	y

things[1]

l	i	k	e	s

things[2]

j	o	k	e	s

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

What About Tuples?

- Basically, a tuple is the same idea as a list, **but** you can't change anything in a tuple after it is created
- To create a tuple you use a pair of parentheses (not square brackets) for the items
- To get a particular item from a tuple it's the same as a list, for example:

friendjohn.py - //VDIDRIVE/MYHOME/rossiter/Documents/friendjohn.py (3.10.4)

File Edit Format Run Options Window Help

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

```
========== RESTART: //
Name of my friend: John
>>>
```

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

Trying to Change an Item in a Tuple

- If you try to change something inside a tuple, the program crashes:

tupletest.py - //VDIDRIVE/MYHOME/rossiter/Documents/tupletest.py (3.10.4)

File Edit Format Run Options Window Help

```
myfriend = ("John", 18, "CSE", 180, 70)
myfriend[2] = "ECE"
```

```
========== RESTART: //VDIDRIVE/MYHOME/rossiter/Documents/tupletest.py ==========
Traceback (most recent call last):
  File "//VDIDRIVE/MYHOME/rossiter/Documents/tupletest.py", line 2, in <module>
    myfriend[2] = "ECE"
TypeError: 'tuple' object does not support item assignment
>>>
```

- Using the language of computing, we say lists are *mutable* (can be changed) and tuples are *immutable* (cannot be changed)

What Works With Tuples?

- `len()`, `count()` and `index()` work because they don't try to change the tuple content
- Creating a 2D structure in a tuple is fine e.g.

```
things = (("h","a","r","r","y"),
          ("l","i","k","e","s"),
          ("j","o","k","e","s"))
```

- `insert()`, `remove()`, `append()`, `sort()`, `reverse()`, `extend()` all **don't work** because they try to change the tuple content

When to Use Tuples

- You may use only lists in your programming because they are ‘more powerful’ than tuples
- However, there are situations where it’s better to use tuples e.g.:
  - You want to ensure some data can’t be changed
  - There are some not common situations where a list doesn’t work but a tuple does, e.g. sometimes in a dictionary (this may be discussed later in another presentation)

A String is a Tuple of Letters

- In computing, a *string* is the name for text
- Python thinks of a string as a tuple of letters
- For example:

stringtest.py - //VDIDRIVE/MYHOME/rossiter/Documents/

File Edit Format Run Options Window Help

```
message = "Hello!"
print(message[1])
```

```
========== RESTART:
e
>>>
```

- Just like a tuple, you can't directly change the content of a string:

stringtest2.py - //VDIDRIVE/MYHOME/rossiter/Docun

File Edit Format Run Options Window Help

```
message = "Hello!"
message[1]="a"
```

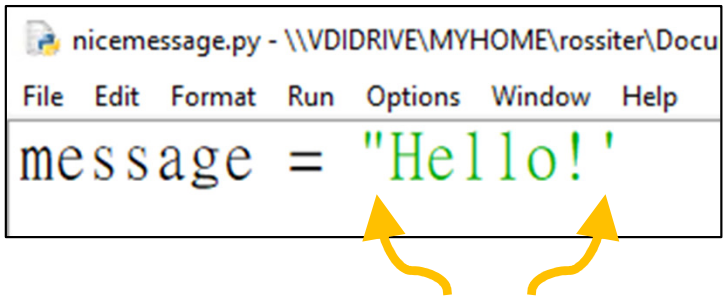
```
========= RESTART: //VDIDRIVE/MYHOME
Traceback (most recent call last):
  File "//VDIDRIVE/MYHOME/rossiter/D
>
    message[1]="a"
TypeError: 'str' object does not sup
```

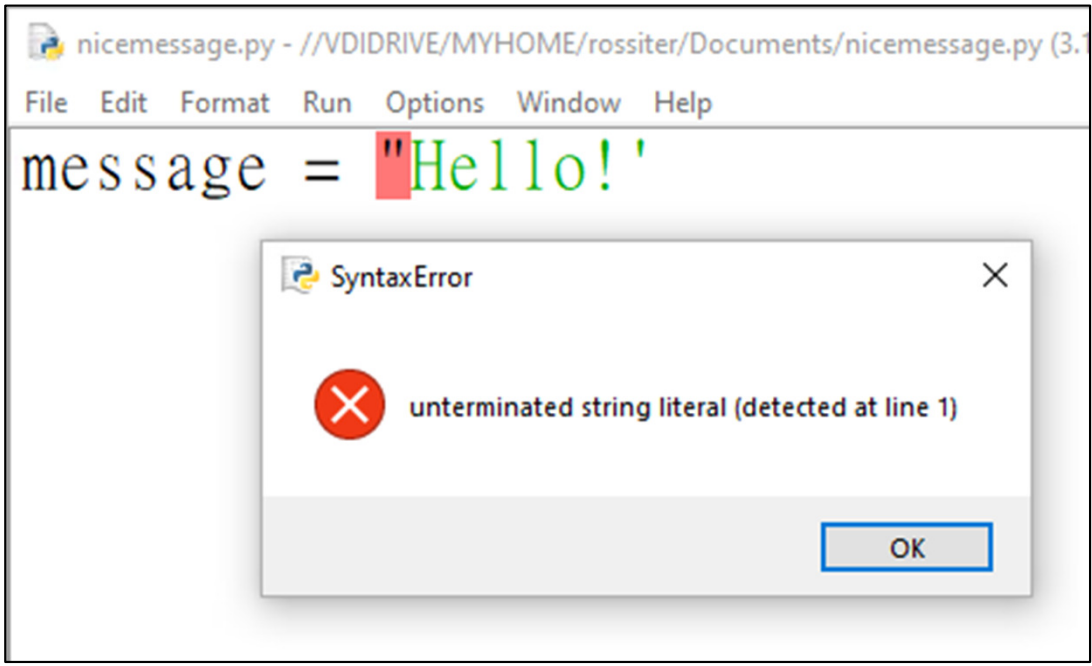
Using Quotes " or '

- As mentioned before, you can use either " " (double quotes) or ' ' (single quotes) for text, for example:

  `'Hello!'` is the same as `"Hello!"`

- However, you cannot mix them, for example:





- Here the start is " but the end is ' - not OK!

What Works With Strings?

- Just like a tuple, with a string you can use `len()`, `count()` and `index()`

stringdemo.py - //VDIDRIVE/MYHOME/rossiter/Documents/stringdemo.py

File Edit Format Run Options Window Help

```
message = "Hello!"
print(len(message))
print(message.count("l"))
print(message.index("o"))
```

```
========== RESTART:
6
2
4
>>>
```

- Just like a tuple, with a string `insert()`, `remove()`, `append()`, `sort()`, `reverse()`, `extend()` all **don't work** because they try to change the string content