

Python and Lists



Admin



- Anyone with questions from last time?
- Calendar:
 - Reminder the dates for the midterm
 - Day Sections: Oct 13 or 14 depending on section
 - No classes on Mon Oct 12 or Wed Nov 11 (State Holidays)
 - Monday schedule on Thursday Oct 15th
- Project?
- <https://medium.com/but-it-works-on-my-machine/2026-software-engineer-trends-retrospective-and-beyond-0323a863f941>
- Reading Assignment: Read chapter 3 in the Python Crash Course Book for this set.

Storing information for later use



- So far how do we store information for later use so far?

Storing information for later use



- So far how do we store information for later use so far?
 - Store the information in variables right?
- What kind of information can we store in variables so far?

Storing information for later use



- So far how do we store information for later use so far?
 - Store the information in variables right?
- What kind of information can we store in variables so far?
 - Strings
 - `name = 'Comp151'`
 - Integers
 - `age = 167`
 - Floating point numbers
 - `gpa = 3.5`

Sometimes we want lots of data



- Sometimes we want to work with lots of data that is related
 - We could just have lots of variables
 - student1 = 'Linus'
 - student2 = 'Cletus'
 - student3 = 'Clement'
 - student4 = 'Sixtus'
 - student5 = 'Cornelius'
 - student6 = 'Cyprian'
 - student7 = 'Lawrence'
 - But that would get tedious **really** quick
 - And “programmers are lazy”
 - Also, we would like those students to be all linked together somehow

Lists: a simple collection



- In python, the simplest **Linear collection** is a **List**
 - **Linear collection:** A data storage mechanism where an arbitrary number of data items are stored with a one by one order
 - There is exactly one first item, exactly one second and so on
- Lists in python can contain any type of data in any position
 - Unlike many languages which must have the same type in all positions
- BUT!!! it is almost always best to keep the same type in all positions in a list
 - So (nearly) all of my examples will be like that.

Creating lists



- Creating an empty list
 - With a **literal** empty list use square brackets
 - `student_list = [ ]`
 - Using the 'factory function'
 - `student_list2 = list()`
- Creating a list with something in it from the start (must be literal)
 - `student_list3 = ["Chrysogonus", "John", "Paul", "Cosmas", "Damion"]`
 - Student list 3 has 5 strings in it

List Items



- Lists start counting from 0
 - Show on board with item numbers
 - Use the `[ ]` operator with the list variable to access a single item in a list
- ```
students = ["Chrysogonus", "John", "Paul", "Cosmas", "Damion"]
student_name = students[2]
```
- This will put “Paul” into the `student_name` variable
  - You can change a list item the same way
  - `students = ["Chrysogonus", "John", "Paul", "Cosmas", "Damion"]`
  - `students[1] = "Enping"`
  - Now `students` has contents:
    - `["Chrysogonus", "Enping", "Paul", "Cosmas", "Damion"]`
-

## List items starting from the last item



- If you want to access a list item starting from the end use negative numbers
  - (lists start from zero when ascending, but -1 when descending)

```
weekdays = ["Sunday", "Monday", "Tuesday", "Wednesday", "Thurbsday", "Friday", "Saturday"]
```

- Let's get that last day

```
last_day = weekdays[-1]
```

- last\_day now contains "Saturday"
- Changing that mistake starting from the end

```
weekdays[-3] = "Thursday"
```

- So now weekdays has the value
  - ['Sunday', 'Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday']

# Notes on Variable names



- Lists hold lots of things
  - So convention says make the name plural
- Where other variables should generally be singular.

## Adding to a list



- If we want to add new items to the list

```
courses = ["comp151", "comp152", "math130", "math120"]
```

- Add at the end of the list

```
courses.append("comp250")
```

- List contents are now
  - ['comp151', 'comp152', 'math130', 'math120', 'comp250']
- Add to a spot in the list
  - If I found that I forgot comp143 and it should be right after comp151

```
courses.insert(1, "Comp143")
```

- The first number is what spot to insert at, the second the thing to insert
- List contents now:
  - ['comp151', 'Comp143', 'comp152', 'math130', 'math120', 'comp250']

## Make sure we do it in code



- If we haven't already, let's make sure to do some of this in code in pycharm
- Also many questions – some of you have them, if we haven't asked yet, please do so now.
  - Computer Code is a different way of thinking, ask now while they are “doubts” rather than letting it snowball to “I have no idea what is going on” in a few weeks.
- Once we do all that – what is the next obvious step in learning lists?

# Removing items from lists



- Removing an item from a list

```
books = ["Python Crash Course", "Game Programming", "Code Complete", "Clean Code", "The Pragmatic Programmer", "Python Crash Course"]
```

- Removing when you know the item to remove (use remove function)

```
books.remove("Python Crash Course")
```

- Remove only removes the first occurrence in the list – so books becomes
  - ['Game Programming', 'Code Complete', 'Clean Code', 'The Pragmatic Programmer', 'Python Crash Course']
- Removing when you know the position to remove (using original list at top of slide with two copies of Python Crash Course)

```
books.pop(2)
```

- Results in the books list being
  - ['Python Crash Course', 'Game Programming', 'Clean Code', 'The Pragmatic Programmer', 'Python Crash Course']
- Remember list “indexes” start from zero

# Index errors



- List Index
  - We refer to the locations in the list using a **list index**
- If you try to use a location that doesn't exist in a list, it is an index error
- So if we have the courses list again

```
courses = ["comp151", "comp152", "math130", "math120"]
```

- How many items are there?
-

# Index errors



- List Index
    - We refer to the locations in the list using a **list index**
  - If you try to use a location that doesn't exist in a list, it is an index error
  - So if we have the courses list again
- ```
courses = ["comp151", "comp152", "math130", "math120"]
```
- How many items are there?
 - Yup 4, and since we start from zero
 - `my_course = courses[4]` will not have anything and will generate an error
 - `Courses[4] = "comp490"` will generate the same error

Reading and Assignment



- Read Chapter 3
 - Though I deferred a couple of the topics from that chapter (sort for example) till later
- Project?