CIS 122 Intro to Programming and Computational Problem Solving

- Python Collections – Lists
  - Lists can be heterogeneous
  - Lists are sequence
  - Lists are mutable

- Python Collections – Tuples

- Intro to Data Ethics
  Guest speaker: Prof. Ramón Alvarado, Philosophy
  (Canvas link)
  PHIL 223 Data Ethics, Spring 2022 (>2)

Data Structures

Data structures are how we store and access data in a computer program.

A data structure
  - organizes data
  - supports basic operations on the data
    (for example, add, update, retrieve, delete)

Python collections – strings, tuples, lists, dictionaries

Python collections – strings, tuples, lists, dictionaries

- flexible collections of data
- modeling real world

Python collections – Sequential

<table>
<thead>
<tr>
<th>Strings</th>
<th>Tuples</th>
<th>Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordered</td>
<td>ordered</td>
<td>ordered</td>
</tr>
<tr>
<td>sequential ops</td>
<td>sequential ops</td>
<td>sequential ops</td>
</tr>
<tr>
<td>characters</td>
<td>multiple types</td>
<td>multiple types</td>
</tr>
<tr>
<td>immutable</td>
<td>immutable</td>
<td>mutable</td>
</tr>
</tbody>
</table>

Lists

>>> frequent_psw = ['abc123', 'password', ...]

Python lists can be heterogeneous

>>> mixitup = ['hello', 'goodbye', True, 99, 99.0, ['a']]
Lists are sequences
so the sequence operators and built in functions
we’ve seen for strings work for lists, too –
[ ], +, *, len, max, min, in, for
also handy list method - count
also handy collections functions – sum, sorted

>>> stuff = [2, 25, 80, 12]
>>> stuff.count(12)
1
>>> sum(stuff)
119
>>> sorted(stuff)
[2, 12, 25, 80]

def all_majorf_multi(name):
    """(name: str) -> list
    Return list of all majors from file,
    where there may be multiple space-separated majors
    on each line of the file.
    >>> all_majors_multi('majors-w22-short-twoperline.txt')
    ['EC', 'MATH', 'CS', 'CIS']
    >>> major = []
    with open(name, 'r') as majorsf:
        majorsf.readline()  # skip the header
        for line in majorsf:
            majors = line.split()  # split on whitespace
            if majors[0] in majors:
                majors.append(major)
    return majors

• Python Collections — Lists
✓ Lists are sequences
✓ Lists can be heterogeneous
• Lists are mutable
• Python Collections — Tuples
• Intro to Data Ethics
Guest speaker: Prof. Ramón Alvarado, Philosophy
PHIL 223 Data Ethics, Spring 2022 (>2)
Lists are mutable – can update in place

```python
>>> mixitup = ['hello', 'goodbye', True, 99, 99.0, ['a']]
>>> mixitup[0] = 'bonjour'
>>> mixitup[1] = 'adios'
>>> mixitup
['bonjour', 'adios', True, 99, 99.0, ['a']]
```

Lists are mutable – can update in place

```python
>>> mixitup = ['hello', 'goodbye', True, 99, 99.0, ['a']]
>>> mixitup[0] = 'bonjour'
>>> mixitup[1] = 'adios'
>>> mixitup
['bonjour', 'adios', True, 99, 99.0, ['a']]
```

compare:
check = 'abc'
check[0] = 'z'

Lists are mutable – can update in place

```python
>>> mixitup[0] = 'bonjour'
>>> mixitup[1] = 'adios'
>>> mixitup
['bonjour', 'adios', True, 99, 99.0, ['a']]
```

Lists are a mutable data type

can change the value of a complex object (including size) during program execution

flexible; powerful; convenient

also

potentially expensive (memory/management)

Python updates the object IN PLACE

Lists are a mutable data type

can change the value of a complex object (including size) during program execution

flexible; powerful; convenient; clear

also

typically expensive (memory management)

>` Python updates the object IN PLACE

Lists are mutable – list methods – be careful!

```python
>>> mixitup
['bonjour', 'adios', True, 99.0, -1, 2, ['a']]
```
Lists are mutable – list methods – be careful!

>>> mixitup
[‘bonjour’, ‘adios’, True, [99.0, -1, 2], [‘a’]]

>>> mixitup = mixitup.append(101)
>>> mixitup

>>> print(mixitup)
None

Lists are mutable – be careful!

>>> list1 = [‘a’, ‘c’, ‘b’]
>>> w1 = ‘ducks’

>>> list1 = list1.sort()
>>> w1 = w1.capitalize()

>>> list1

>>> w1

>>> list1 = [‘a’, ‘c’, ‘b’]
>>> list1 = sorted(list1)

>>> list1

Recall: And for lists:

>>> b = 20
>>> y = [1, 2, 3]
>>> a = b
>>> x = y

>>> b = 30
>>> y = [4, 5, 6]

>>> b >>> a

30 20

?? ??
Recall:

Now for list updated in place:

```python
>>> b = 20
>>> y = [1, 2, 3]
>>> a = b
>>> x = y
>>> b = 30
>>> y[1] = 99
>>> b
>>> a
>>> y
>>> x
```

20
30
?
?

```python
30
```

30
20
??
??

**aliasing** (occurs when one variable is assigned to the value of another var): two names for one object

make a copy of the object to avoid aliasing

```python
be_careful(li):
    return li.append(99)
```

```python
myl = [1, 2, 3]
be_careful(myl)
myl
```

No → tuple – safer and faster

Yes → list

Is a list the best choice for representing data?

Does the data need to be changed?

Lists – heterogenous, mutable – are a very flexible and powerful data type: use wisely!
• Python Collections – Lists
  ✓ Lists are sequences
  ✓ Lists can be heterogeneous
  ✓ Lists are mutable

• Python Collections – Tuples

• Intro to Data Ethics
  Guest speaker: Prof. Ramón Alvarado, Philosophy
  PHIL 223 Data Ethics, Spring 2022 (>2)

Python collections – Sequential

<table>
<thead>
<tr>
<th>Strings</th>
<th>Tuples</th>
<th>Lists</th>
</tr>
</thead>
<tbody>
<tr>
<td>ordered</td>
<td>ordered</td>
<td>ordered</td>
</tr>
<tr>
<td>sequential ops</td>
<td>sequential ops</td>
<td>sequential ops</td>
</tr>
<tr>
<td>characters</td>
<td>multiple types</td>
<td>multiple types</td>
</tr>
<tr>
<td>(including lists)</td>
<td>(including lists)</td>
<td></td>
</tr>
<tr>
<td>immutable</td>
<td>immutable</td>
<td>mutable</td>
</tr>
</tbody>
</table>

Python tuples

For example,
```
int_seq = (10, 20, 30, 40, 50)
gen_seq = (10, 20.0, 'a', True)
nested_seq = (10, 20, ('a', 'b'), True)
```

```
>>> int_seq[0]
10
>>> len(nested_seq)
3
>>> int_seq[1:4]
(20, 30, 40)
```

Python tuples

For example,
```
>>> int_seq = (10, 20, 30, 40, 50)
>>> gen_seq = (10, 20.0, 'a', True)
>>> nested_seq = (10, 20, ('a', 'b'), True)
```

```
>>> a = nested_seq[2][0]
>>> x = 10,000
>>> b = nested_seq[2][1]
>>> (a, b) = (a + 2, b + 2)
```

```
>>> a, b, c, d, e = int_seq
>>> short_seq = (99,)
```

```
>>> a, b = b, a
```

```
A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122 Programming concepts so far
Computational problem solving (from problem to algorithm to implementation) process to program to high quality program

CIS 122 Intro to Programming and Computational Problem Solving

Python collections – Sequential – Index access

Lists – heterogeneous, mutable – are a very flexible and powerful data type: use wisely!

Is a list the best choice for representing data?

Does the data need to be changed?

No → tuple – safer and faster
Yes → list

CIS 122 Intro to Programming and Computational Problem Solving

Python Collections – Lists
✓ Lists are sequences
✓ Lists can be heterogeneous
✓ Lists are mutable

Python Collections – Tuples

Intro to Data Ethics
Guest speaker: Prof. Ramón Alvarado, Philosophy
PHIL 223 Data Ethics, Spring 2022 (>2)

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

A Structured Approach to Computational Problem Solving
CIS 122 is a community of learners where

- Everyone is welcome
- Everyone is respected
- We value intellectual challenges and deliberate practice in pursuit of new knowledge and skills
- We support and encourage each other
- We celebrate our own and each other’s accomplishments

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

What can you expect in CIS 122?

✓ Weekly projects and exercises to support learning of computational problem solving in a variety of areas

Supported by

✓ Class – large group – big picture concepts, exercises, Q/A
✓ Lab – small group – exercises, computers/whiteboards, Q/A
✓ Class notes (posted), project solutions (posted), text readings – review, practice, explore
✓ Help hours – daily drop-in help, tutorials, code reviews
✓ Comprehensive assessments – 2 midterm + 1 final “demo”
✓ Individual feedback on weekly projects

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

Welcome to CIS 122
Intro to Programming and Computational Problem Solving

CIS 122 Learning Outcomes

- use a computational problem-solving approach to generate computer solutions (programs) to a variety of problems
- write well-structured, well-documented programs using the Python programming language
- be prepared to continue to study programming or computer science on your own or in other courses