

# Structured Types

Guttag Chapter 5

# Structured Types Roadmap

Python has build-in data types

- int, float, bool
- Tuple, List, str
- set, dictionary

Assignment on Structured Types (Specifications Lab) and Intersection Algorithms (Engineering Lab) will be out Wednesday, due next Wed and Thurs respectively.

# Goals for today

## Types

- Tuples
- Lists
- str

## Terms

- mutable vs immutable
- indexable
- ordered
- slicing

## Algorithms

- Intersection (using tuples)
- Apply to Each (using lists)

# Tuples

# Tuples contain values of any type

```
# these are tuples
new_pair = (3.2, 4)
new_quadruple = ("Story number", 3, "is", True)
```

```
# these are tuples
pair = 3, 4.9
quadruple = "Story number", 3, "is", False
```

```
# these are tuples
tuple_empty = ()
tuple_str = ("Story",)
tuple_int = (3,)
tuple_float = (3.14159,)
```

```
# these are NOT tuples
example_int = (100)
example_float = (100.001)
```

## Adding Tuples Creates Entirely New Tuple

```
pair = (3.2, 4)
quadruple = ("Story number", 3, "is", True)
new_tuple = pair + quadruple
print(new_tuple)
```

```
(3.2, 4, 'Story number', 3, 'is', True)
```

# Tuples are indexable using []

```
a = (1, 2, 3)  
print(a[0])
```

what prints, what is type(a)?

```
a = (1, 2, 3)[0]
```

what is type(a)?

# Tuples are Immutable

```
a = (1, 2, 3)
a[0] = 10
```

**CRASH with attempted reassignment**

---

```
TypeError                                Traceback (most recent call last)
<ipython-input-1-8b43e861f98c> in <cell line: 4>()
      2
      3 a = (1, 2, 3)
----> 4 a[0] = 10
      5
      6 # is `a` a tuple?
```

TypeError: 'tuple' object does not support item assignment



# Tuple Summary

As you can see...

- tuples are storage containers
- tuples can contain mixed values
- addition creates a new tuple
- tuples are indexable, elements can be accessed with []
- tuples are immutable, cannot be changed after creation

See python notebook on tuples

# Lists

## Lists contain values of any type

```
# these are lists
```

```
new_pair = [3.2, 4]
```

```
new_quadruple = ["Story number", 3, "is", True]
```

```
# these are lists
```

```
list_empty = []
```

```
list_str = ["Story"]
```

```
list_int = [3]
```

```
list_float = [3.14159]
```

## Adding Lists Creates Entirely New List

```
pair = [3.2, 4]
quadruple = ["Story number", 3, "is", True]
new_list = pair + quadruple
print(new_list)
```

```
[3.2, 4, 'Story number', 3, 'is', True]
```

Lists are indexable using []

```
a = [1, 2, 3]  
print(a[0])
```

what prints, what is type(a)?

```
a = [1, 2, 3][0]
```

what is type(a)?

## List are Mutable


```
a = [1, 2, 3]  
a[0] = 10
```

**RUNS with reassignment**

# Appending to an Existing List

```
a = [1, 2, 3]  
a.append(1)  
print(a)
```

"dot" notation, no equals sign  
because appending modifies  
an existing list



```
[1, 2, 3, 1]
```

```
a = [1, 2, 3]  
list.append(a, 1)  
print(a)
```

```
[1, 2, 3, 1]
```

# List Summary

As you can see...

- lists are storage containers
- lists can contain mixed values
- addition creates a new list
- lists are indexable, elements can be accessed with []
- lists are mutable, can change after creation
  - by overwriting values at certain index
  - by appending to end of the list

See python notebook on lists



# Strings

## Strings Contain Characters

```
# strings contain characters
```

```
a = 'this is a string'
```

```
digit_string = '1234567890'
```

```
another_string = '1, 2, 3, 4, 5, 6, 7, 8, 9, 0'
```

```
empty_string = ''
```

# Strings are immutable

```
# strings are immutable  
a = 'this is a string'  
a[0] = 'T'
```

---

```
-----  
TypeError                                Traceback (most recent call last)  
<ipython-input-2-71d026a178ed> in <cell line: 3>()  
      1 # strings are mutable  
      2 a = 'this is a string'  
----> 3 a[0] = 'T'
```

TypeError: 'str' object does not support item assignment

Strings are indexable with []

```
# strings are indexable, ordered  
a = 'this is a string'  
print(a[0])  
print(a[1])
```

t  
h

## Adding Strings Creates Entirely New String

```
# adding creates an entirely new string  
a = 'this is a string'  
another_string = '1, 2, 3, 4, 5, 6, 7, 8, 9, 0'  
new_string = a + another_string  
print(new_string)
```

this is a string1, 2, 3, 4, 5, 6, 7, 8, 9, 0

# String Summary

As you can see...

- strings contain characters only
- addition creates a new string
- strings are indexable, elements can be accessed with []
- strings are immutable, cannot be changed after creation

See python notebook on strings

# Tuples vs Strings vs Lists

# Tuples and Strings

- New tuple with ( ), new string with ""
- Indexable, ordered
- Immutable

Need to change something?

- Make a new one using addition
  - new memory chunk needed
  - every element has to be copied

# Lists

- New list with [ ]
- Indexable, ordered
- Mutable

Need to change something?

- Make a new list using addition
  - new memory chunk needed
  - every element has to be copied
- Changing existing list
  - appending - new element is simply added onto the end of items in the memory chunk containing the list
  - indexing in and overwriting



# Speed

## Tuple

- The python interpreter can access data faster from a tuple

## List

- If the values in the container have to update a lot, a list is faster because appending (list only) does not require copying values into new memory locations

## Strings

- Not used as a data container since they only contain characters

# Slicing

# Slicing

applies to any type that is indexable

- lists, tuples, strings

indices given  
with integer  
index!

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]
# retrieve elements 0 up to 3rd (not including 3rd)
print(a[0:3])
```

```
[0, 'elem1', 'elem2']
```

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]
# retrieve elements 0 up to 8th (not including 8th), in step size of 2
print(a[0:8:2])
```

```
[0, 'elem2', 'this is the fourth element', '6.0000']
```

## Slicing to the end

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]  
# retrieve elements 5 up to 8th (including 8th)  
print(a[5:])
```

```
['5', '6.0000', 7, 8.0]
```

# Slicing backward

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]  
# retrieve elements from end to 0 (not including 0) backward (step size -1)  
print(a[:0:-1])
```

```
[8.0, 7, '6.0000', '5', 'this is the fourth element', 3, 'elem2', 'elem1']
```

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]  
# retrieve elements from end to 0 (including 0) backward (step size -1)  
print(a[::-1])
```

```
[8.0, 7, '6.0000', '5', 'this is the fourth element', 3, 'elem2', 'elem1', 0]
```

## Slicing with reference to the end

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]  
# retrieve elements 2 up to 2nd to last (not including 2nd to last)  
print(a[2:-2])
```

```
['elem2', 3, 'this is the fourth element', '5', '6.0000']
```

# Attendance question from today

What will this print?

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]  
# retrieve a string and do further indexing  
print(a[4][8:-8])
```

## pretty\_print\_list

```
def pretty_print_list(values: Iterable[int]) -> str:
```

```
    """Pretty print a list without brackets and adding commas."""
```

```
from typing import Iterable
```

```
def prettyprint(values: Iterable[int]) -> str:  
    return str(values)[1:-1]
```



# Iterating through list or tuple or string or range

```
a = [0, 'elem1', 'elem2', 3, 'this is the fourth element', '5', '6.0000', 7, 8.0000]

for item in a:
    print(item)
```

# Explore Tuples, Lists, Strings and Slicing

Tuples:

[https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007\\_structured\\_types\\_tuples.ipynb](https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007_structured_types_tuples.ipynb)

Lists:

[https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007\\_structured\\_types\\_lists.ipynb](https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007_structured_types_lists.ipynb)

Strings and

Slicing: [https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007\\_strings\\_and\\_slicing.ipynb](https://github.com/allegHENY-college-cmpsc-101-fall-2024/course-materials/blob/main/code/20241007_strings_and_slicing.ipynb)

Assignment on Structured Types (Specifications Lab) and Intersection Algorithms (Engineering Lab) will be out Wednesday, due next Wed and Thurs respectively.