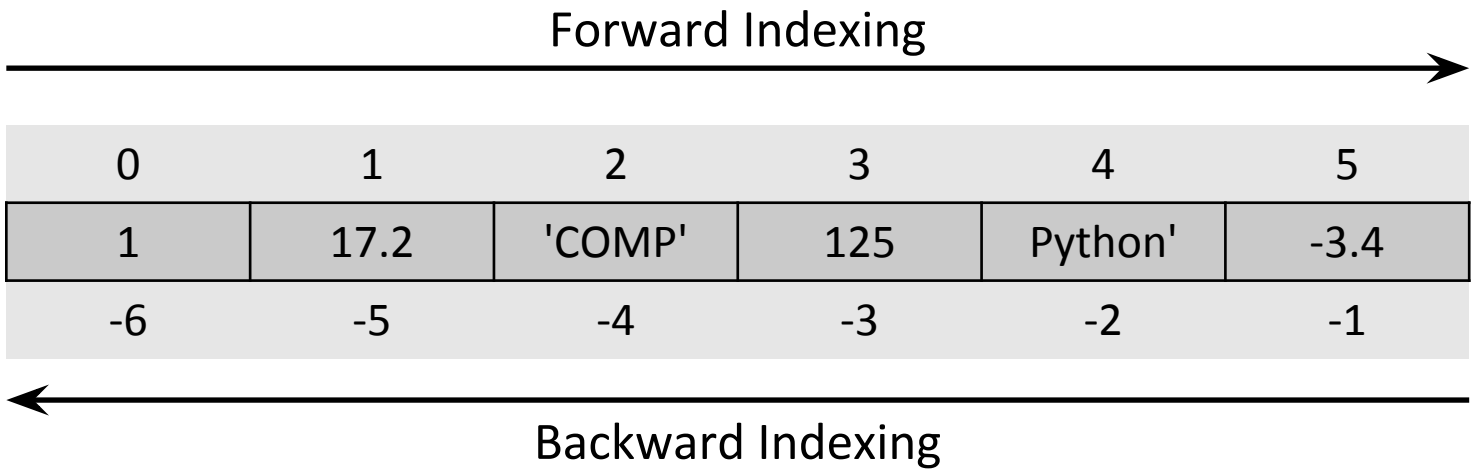
COMP 125 Programming with Python
Tuples



This but immutable

Mehmet Sayar

Koç University

Strings vs Lists

- `len(), print()`
- slicing, indexing
- foreach loops
- `in`
- concatenation
- Immutable
  - Can’t add or remove elements

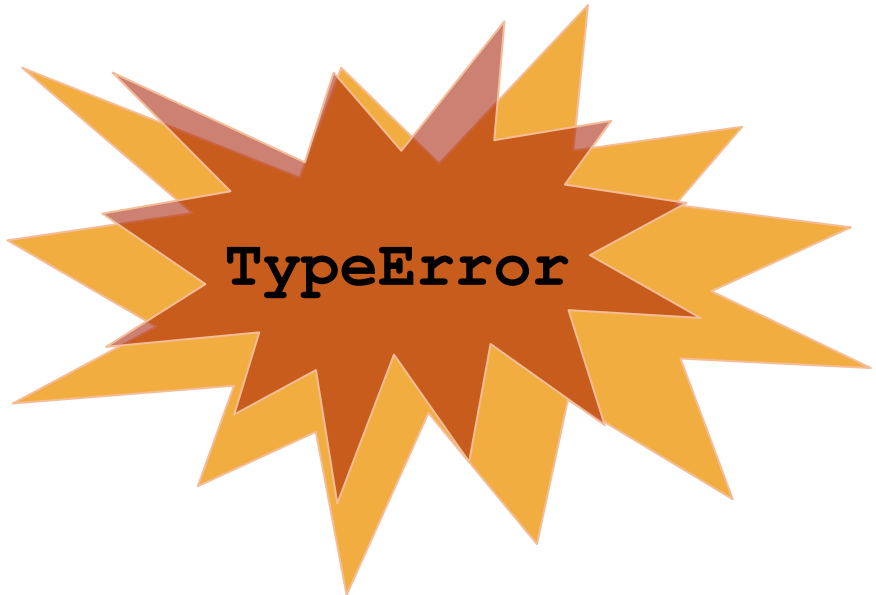
- `len(), print()`
- slicing, indexing
- foreach loops
- `in`
- concatenation
- mutable
  - `append()`
  - `pop()`
  - Assign with indexing
  - `del item`

Mutability

- A mutable object can be changed after it's created.
- An immutable object can't.

```
lst = [1, 2, 3]
lst[0] = 'a'
lst → ['a', 2, 3]
```

```
s = '123'
s[0] = 'a'
```



We are going to discuss the benefits of immutability and mutability as they come up.

Tuples

- “bundles of small amount of data”: An immutable data type for storing values in an ordered linear collection.
- In other words, immutable lists
- Use parentheses (( )) instead of brackets ([ ])to define them. E.g.

```
('a', 'b', 'c')
('karel', 1)
('simba', 'lion', 25)
```

Tuples vs Lists

- `len(), print()`
- slicing, indexing
- foreach loops
- `in`
- Concatenation
- Store arbitrary elements
- Immutable
  - Can’t add or remove elements

- `len(), print()`
- slicing, indexing
- foreach loops
- `in`
- Concatenation
- Store arbitrary element
- mutable
  - `append()`
  - `pop()`
  - Assign with indexing
  - `del item`

Tuples don’t support item assignment

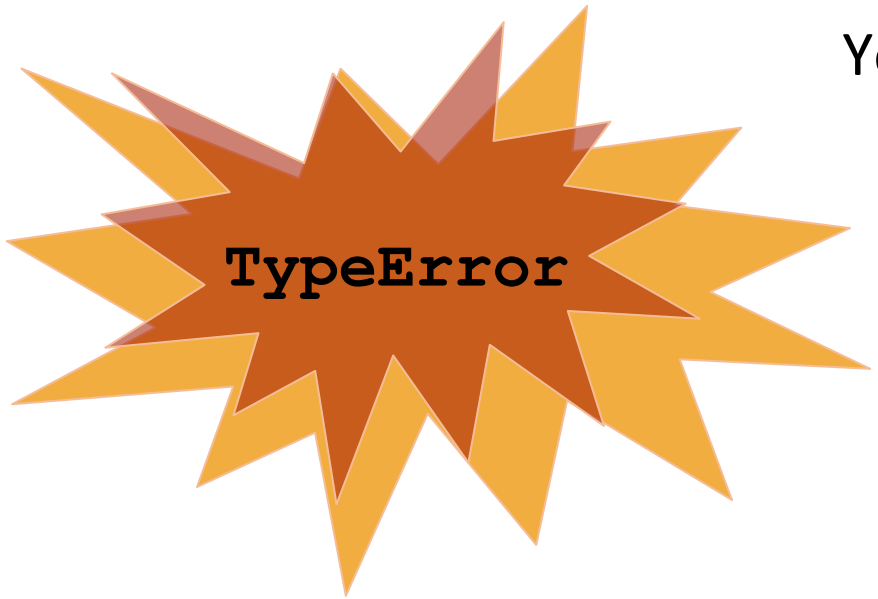
tup = ('apple', 0.79, 'WA')

a = tup[0] ⟶ 'apple’

a ⟶ 'apple’

You can index in to view the elements

tup[2] = 'CA'



You **can’t** index in to **set** one of the elements

No Parentheses Needed!

- You do not need parentheses to create tuples

```
tuple1 = (True, -3.4, 'hello', 1)
tuple2 = True, -3.4, 'hello', 1
print(tuple1)
print(tuple2)
```

Output:

```
(True, -3.4, 'hello', 1) <class 'tuple’>
(True, -3.4, 'hello', 1) <class 'tuple’>
```

- However, you should when creating a new one! It is better for **readability**

Packing and Unpacking

- Packing: Assigning multiple values (or an iterable) to a tuple
  - The no parentheses example

```
tup = 'elma', 7.99, 'market'
```

  - Possible to do packing with a list. We will see more later on

- Unpacking: Assigning the contents of the tuple to multiple values

```
urun, fiyat, satici = tup
Urun → 'elma'
```

  - Other iterables can also be unpacked!

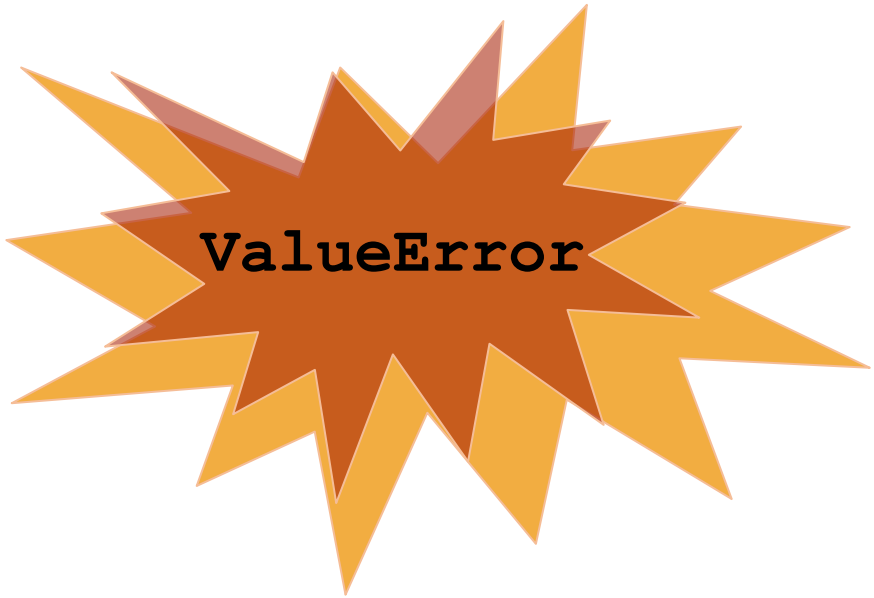
Detour: Unpacking with Other Iterables

- Working:

```
a, b, c = [1,2,3]   | a → 1,   b → 2,   c → 3
a, b, c = '123’     | a → '1', b → '2', c → '3’
a, b, c = range(3)  | a → 0,   b → 1,   c → 2
```

- Non-working: Length of the left-hand side must be equal to the length of the right-hand side

```
a, b, c, d = [1,2,3]
a, b = [1,2]
```



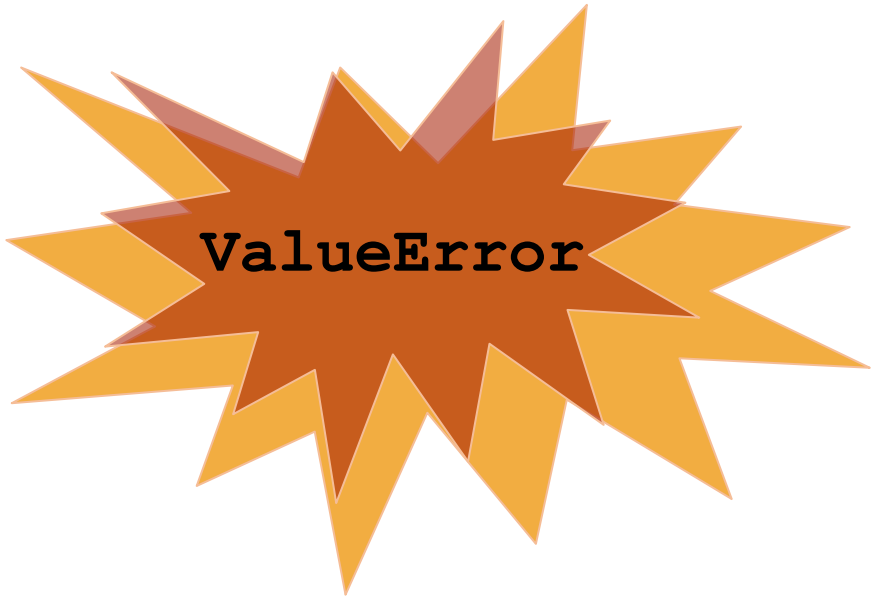
More Unpacking with Tuples

- Working

```
a, b, c = 1, 2, 3     | a → 1, b → 2, c → 3
a, b, c = (1, 2, 3)   | a → 1, b → 2, c → 3
a, b, c = 1, 2, 3     | a → 1, b → 2, c → 3
```

- Non-working: Length of the left-hand side must be equal to the length of the right-hand side

```
a, b = 1, 2, 3
a, b, c = 1, 2
```



Swapping

```
a, b = 1, 2 | a → 1, b → 2
```

- Swapping is commonly needed for programming. Python's unpacking feature makes this very elegant!

```
a, b = b, a | a → 2, b → 1
```

Tuple Operations: Similar to Lists

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!',) * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Tuple Operations: Unlike Lists

- Tuples do not support the methods:
  - `append`
  - `remove`
  - `insert`
  - `reverse`
  - `sort`
- Tuples:
  - Fixed number of elements, need to know ahead of time!
  - Usually used to store small number of elements
- Lists:
  - Unbounded (memory permitting) number of elements
  - Preferred to sotre large number of elements

Pick: Tuple or Lists

- store many website urls → list
- store x, y, z coordinates together → tuple
- store many pixels → list
- store “name” and student id together → tuple
- store filename of mp3 file and its length in seconds → tuple

Tuples

- Advantages for using tuples over lists:
  - Processing tuples is faster than processing lists
  - Tuples are safe (immutable)
  - Some operations in Python require use of tuples (we will see why)

- `list()` function: converts tuple to list
- `tuple()` function: converts list to tuple

- Tuples are frequently used to “name” values as well

```
prices = [('elma', 7.99), ('ayva', 8.95)]
```

Sorting Lists with Tuples

```
lst = [('mango', 3), ('apple', 6), ('lychee', 1), ('apricot', 10)]
print(sorted(lst))
```

Output:

```
[('apple', 6), ('apricot', 10), ('lychee', 1), ('mango', 3)]
```

Sorts by the first element in each tuple