

1. Recap
○○
○○○

2. Data Model
○○

3. Aliasing & Cloning
○○

4. Mutability
○

5. Tuples
○○

6. Sets
○○○

7. Dictionaries
○○○

KOLT Python

Containers, Aliasing & Mutability

Ahmet Uysal

Tuesday 3rd March, 2020



**KOÇ
UNIVERSITY**

OFFICE OF LEARNING AND TEACHING



1. Recap
○○
○○○

2. Data Model
○○

3. Aliasing & Cloning
○○

4. Mutability
○

5. Tuples
○○

6. Sets
○○○

7. Dictionaries
○○○

Agenda

Functions

Lists

1. Data Model

2. Aliasing & Cloning

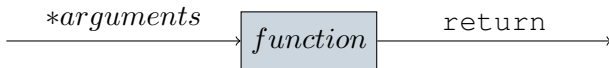
3. Mutability

4. Tuples

5. Sets

6. Dictionaries

Functions



```
def function_name(parameter2, parameter2, ...):  
    <expression>  
    ...  
    return value
```

```
fib_100 = fibonacci_series(100)  
what_is_going_on = print(fib_100)
```

return Statement

Every function returns one value!
What type does each function return?

```
def square(x):  
    return x**2
```

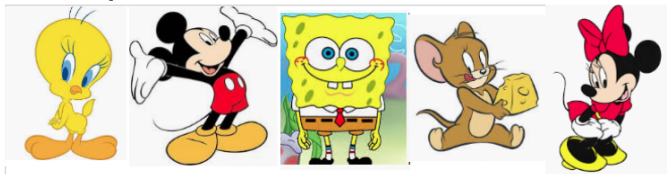
```
def your_full_name(name, surname):  
    return name + ' ' + surname
```

```
def what_is_the_meaning_of_life(life):  
    print("I guess it's nothing")
```

```
def who_are_my_instructors(student):  
    instructors = ['Ahmet', 'Ceren', 'Gül Sena', 'Hasan Can']  
    return instructors
```

Sponge Bob seeks for Sandy

```
cartoon_characters=['Tweety', 'Mickey', 'Sponge Bob', 'Jerry',  
'Minnie']
```



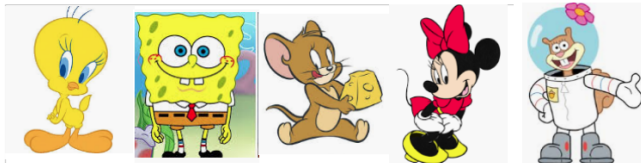
```
cartoon_characters.append('Sandy')
```



Let's play

But, what good is Mickey without being near to Minnie?

```
cartoon_characters.remove('Mickey')
```



0

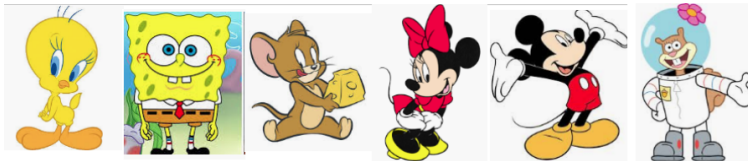
1

2

3

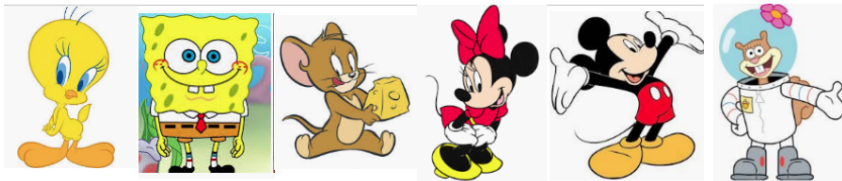
4

```
cartoon_characters.insert(4, 'Mickey')
```



List Operations

Be quick!



```
len(cartoon_characters) ⇒ 6
```

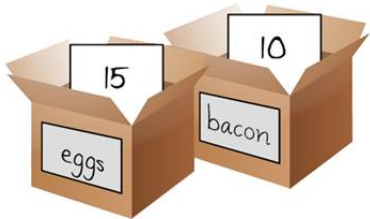
```
cartoon_characters[6] ⇒ Error
```

```
'Jerry' in cartoon_characters ⇒ True
```

```
cartoon_characters.index('Tweety') ⇒ 0
```

Python Data Model

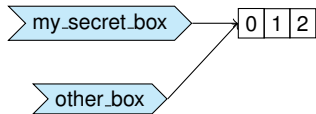
How did we represent data in Python? **Variables!**
How do they work? Do they store the data themselves?



NO! Variables point to Python objects

Python Data Model

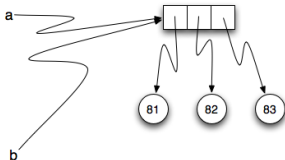
```
my_secret_box = [0, 1, 2]  
other_box = my_secret_box  
other_box.remove(2)  
print(my_secret_box)
```



Variables are more like **labels** pointing to **values**!
Assignment links **variables** to **values**!

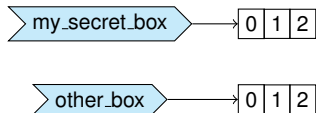
Aliasing & Cloning

- More than one variables can refer to **same object**!
- This is known as **aliasing**, i.e, having more than one name.
- What if we want to clone/copy instead of aliasing?
- For lists, `list.copy()` \Rightarrow returns a shallow copy of the list.
- Shallow: only copy the references, not inner values.



Cloning

```
my_secret_box = [0, 1, 2]
other_box =
my_secret_box.clone()
other_box.remove(2)
print(my_secret_box)
```



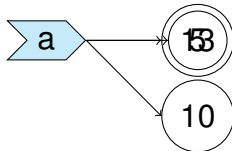
Mutability

Immutable:

An **object** with a fixed value.

- **numbers, strings, tuples, ...**
- Such an object cannot be altered
- A new object has to be created if a different value has to be stored

```
a = 5  
a = 10  
a += 3  
print(a)
```



Tuples

- **Immutable** sequence(ordered) of elements.
- Similar to `lists`, you can use **indexing**, **slicing**, and iterate over using `for` loops.
- Elements cannot be added/removed/changed once the tuple is created.
- How to create tuples? `my_tuple = (1, [1, 2], 'a')`
- `len(my_tuple) ⇒ 3`
- `my_tuple.append(3) ⇒ AttributeError:`
`'tuple' object has no attribute 'append'`

Tuples

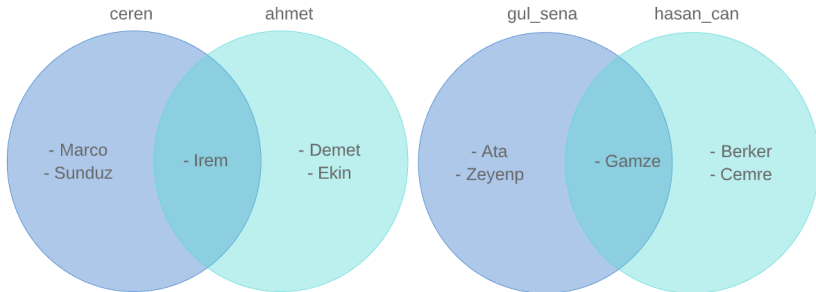
() / tuple(): empty tuple,
(3): int 3,
(3,): tuple containing 3

```
my_list = [1, 2, 3]
my_tuple = ('a', my_list) # ('a', [1, 2, 3, 4])
my_list.append(4)
print(my_tuple)
my_list += [5, 6, 7] # my_list.extend(...)
print(my_tuple)
my_tuple += (1, 2) # my_tuple = my_tuple + (1, 2)
print(my_tuple)
```

Sets

- **Unordered** sequence of **unique** elements.
- **Cannot** use **indexing/slicing**, can iterate with `for` loops.
- **Mutable**, `add(element)`, `remove(element)` methods.
- Python also has **immutable** sets: `frozenset`
- How to create sets? `my_set = {1, 2, 3, 4, 2}`
- How to create empty sets? `set()` (`{ }` is reserved for `dict`)
- Can compute set operations: **union**, **intersection**, **difference**, **symmetric difference**.

Sets



Sets

```
ceren = {'Marco', 'Irem', 'Sunduz'}
gul_sena = {'Gamze', 'Ata', 'Zeynep'}
hasan_can = {'Gamze', 'Berker', 'Cemre'}
ahmet = {'Irem', 'Demet', 'Ekin'}

# intersection &
print(gul_sena.intersection(hasan_can)) # => {'Gamze'}
print(ceren & gul_sena) # => set()
# union |
print(ceren.union(ahmet)) # => {'Ekin', 'Irem', 'Demet',
                                # 'Marco', 'Sunduz'}
print(hasan_can | ceren | gul_sena | ahmet) # => all names
# difference -
print((gul_sena - hasan_can)) # => {'Zeynep', 'Ata'}
# symmetric_difference ^
print(ceren.symmetric_difference(ahmet))
# => {'Marco', 'Ekin', 'Sunduz', 'Demet'}}
```

Dictionaries

- Collection of **key–value** pairs.
- **Cannot** use **indexing/slicing**, can iterate with `for` loops.
- In general, they are not **ordered**.
- However, in Python 3.7 pairs are guaranteed to be in insertion order.
- In other words, we will get pairs in insertion order if we loop over the `dict`.
- How to create dictionaries? `{ }/dict()`: empty dictionary
- `d = {'one': 1, 'two': 2, 'three': 3, 'four': 4}`
- How to access values? `print(d['one'])` `# ⇒ 1`

Confused Section Leader Gul Sena

```
# I need a way to keep track of my students
my_students = {'Ayse': ['economics', 'freshman'],
               'Emir': ['psychology', 'master'],
               'Emirhan': ['business administration', 'junior'],
               'Furkan': ['law', 'junior'],
               'Mahsa': ['material science', 'phd'],
               'Meva': ['international relations', 'freshman']}

for student, info in my_students.items():
    print(f'{student} studies {info[0]}')
# Emir left my class :(
my_students.pop('Emir')
# someone new in my class
my_students['Canan'] = ['industrial engineering', 'junior']
# Ayse passed another year
my_students['Ayse'][1] = 'sophomore'
```

Announcements

Fill out the attendance form:

tiny.cc/koltpython

Keyword: **ceren**

Assignment I: Tic-Tac-Toe is due tonight!

