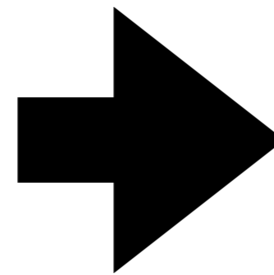


[320] Object Oriented Programming

Meenakshi Syamkumar

Creating New Types

CLASSES AND OTHER TYPES



OBJECTS



Classes


dicts can represent many kinds of things

classes (today) are often a better option
when all your keys are the same

```
m1 = {...}
```

```
m2 = {...}
```

create some objects
of type **dict** for **movies**




```
p1 = {}
```

```
p2 = {}
```

```
p3 = dict()
```

create some objects
of type **dict** for **people**



set some keys/values



```
p1["Fname"] = "Joseph"
```

```
p2["fname"] = "Peyman"
```

```
p3["fname"] = "Shri Shruthi"
```

```
print(type(m1))
```

```
print(type(p1))
```

```
class Person:  
    pass
```

← create a Person
type/class

```
p1 = Person()  
p2 = Person()  
p3 = Person()
```

← create some objects
of type Person

```
p1.fname = "Joseph"  
p2.fname = "Peyman"  
p3.fname = "Shri Shruthi"
```

← set some attributes

```
print(type(p3))
```

Objects created from classes are mutable.
Attribute names are not fixed at creation.

PythonTutor: Compare dicts to class types

```
Python 3.6  
known limitations  
-----  
1 p1 = {"x": 4, "y": 5}  
2  
3 class Coord:  
4     pass  
5  
6 p2 = Coord()  
7 p2.x = 4  
→ 8 p2.y = 5  
-----
```

[Edit this code](#)

→ line that just executed

→ next line to execute



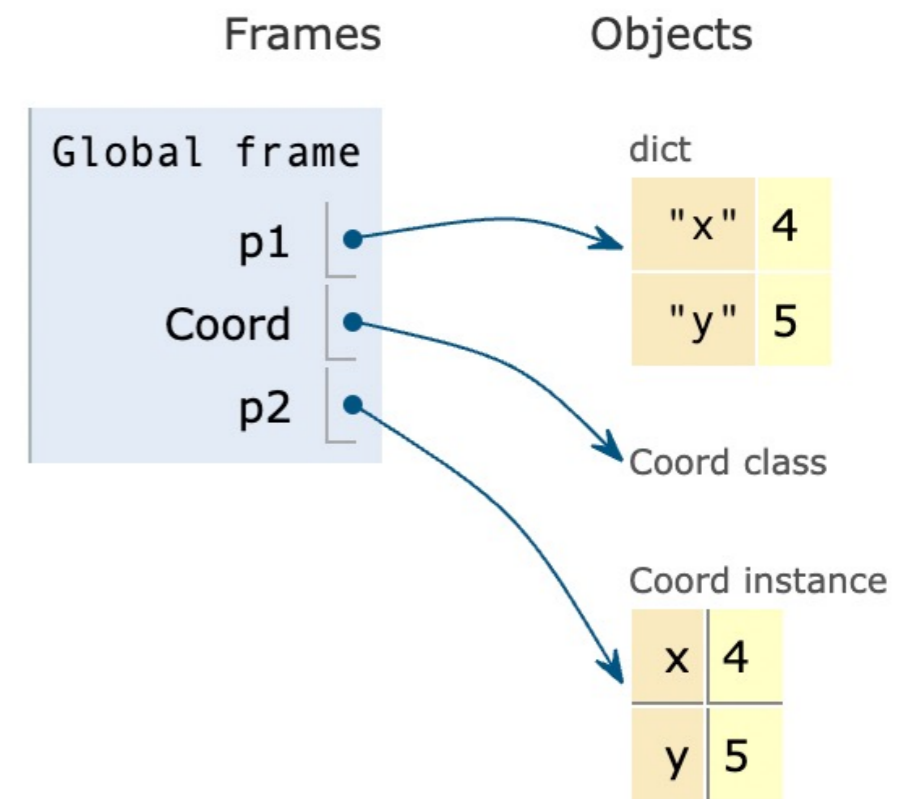
<< First < Prev Next > Last >>

Done running (5 steps)

Visualized with pythontutor.com

NEW: [subscribe](#) to our YouTube

[Move and hide objects](#)



Coding Examples: Animal Classes

Principals

- objects and functions
- methods
- checking object type
- type-based dispatch
- receiver (self parameter)
- constructors

