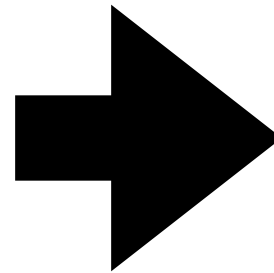


# [320] Object Oriented Programming

Tyler Caraza-Harter

# Creating New Types

## CLASSES AND OTHER TYPES



## OBJECTS



# Classes


**dicts** can represent many kinds of things

[https://earthquake.usgs.gov/fdsnws/event/1/query?  
format=geojson&starttime=2021-09-21&endtime=2021-09-22](https://earthquake.usgs.gov/fdsnws/event/1/query?format=geojson&starttime=2021-09-21&endtime=2021-09-22)

**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**



```
p1["Fname"] = "Joseph"  
p2["fname"] = "Peyman"  
p3["fname"] = "Shri Shruthi"
```

set some keys/values



```
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))
```

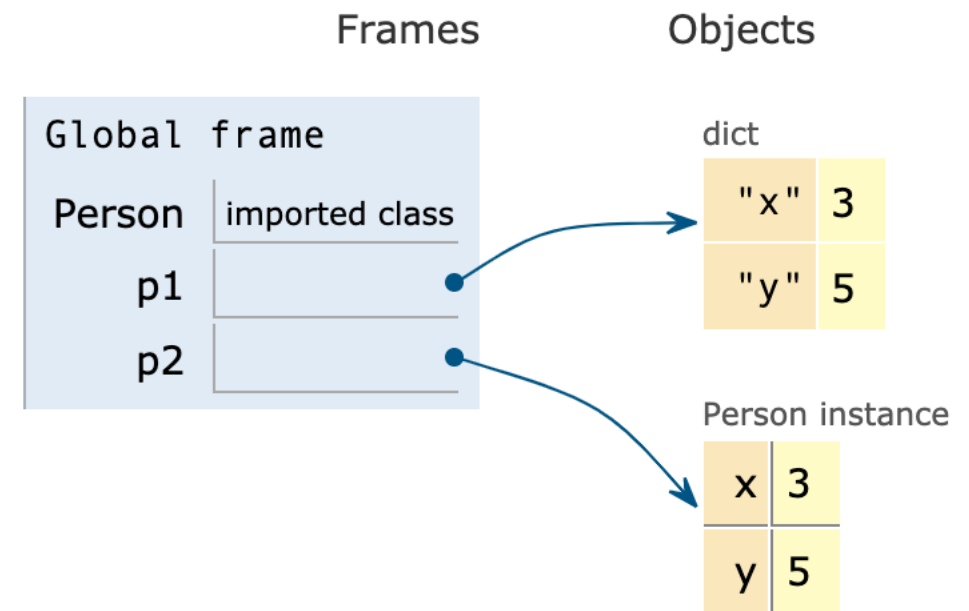
Objects created from classes are mutable.  
Attribute names are not fixed at creation.

# PythonTutor: Compare dicts to class types

Write code in Python 3.6 ▼

(drag lower right corner to resize code editor)

```
1 class Person():
2     pass
3
4 p1 = dict()
5 p1["x"] = 3
6 p1["y"] = 5
7
8 p2 = Person()
9 p2.x = 3
10 p2.y = 5
```





# Coding Examples: Animal Classes

## Principals

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

