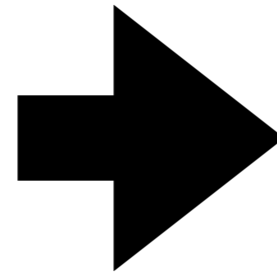


[320] Object Oriented Programming

Tyler Caraza-Harter

Creating New Types

CLASSES AND OTHER TYPES




OBJECTS



Class Attributes

```
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.

Attribute Names/Values are like Keys/Values

USING DICT

```
d = dict()
```

```
d["x"] = 3  
d["y"] = 4
```

```
tot = d["x"] + d["y"]
```

```
has_z = "z" in d
```

USING

```
class Point:  
    pass
```

```
p = Point()
```

```
setattr(p, "x", 3)  
setattr(p, "y", 4)
```

```
tot = (getattr(p, "x")  
       + getattr(p, "y"))
```

```
has_z = hasattr(p, "z")
```

```
p = Point()
```

```
p.x = 3  
p.y = 4
```

```
tot = p.x + p.y
```

```
# no equivalent
```

Attribute Names/Values are like Keys/Values

USING DICT

```
d = dict()
```

```
d["x"] = 3  
d["y"] = 4
```

```
tot = d["x"] + d["y"]
```

```
has_z = "z" in d
```

USING

```
class Point:  
    pass
```

```
p = Point()
```

```
setattr(p, "x", 3)  
setattr(p, "y", 4)
```

```
tot = (getattr(p, "x")  
       + getattr(p, "y"))
```

```
has_z = hasattr(p, "z")
```

avoid this

```
p = Point()
```

```
p.x = 3  
p.y = 4
```

```
tot = p.x + p.y
```

```
# no equivalent
```

preferred

only use attribute
names that could also
be variables names

Attribute Names/Values are like Keys/Values

USING DICT

```
d = dict()
```

```
d["x"] = 3  
d["y"] = 4
```

```
tot = d["x"] + d["y"]
```

```
has_z = "z" in d
```

USING

```
class Point:  
    pass
```

```
p = Point()
```

```
setattr(p, "x", 3)  
setattr(p, "y", 4)
```

```
tot = (getattr(p, "x")  
       + getattr(p, "y"))
```

```
has_z = hasattr(p, "z")
```

avoid this

```
p = Point()
```

```
p.x = 3  
p.y = 4
```

```
tot = p.x + p.y
```

```
# no equivalent
```

preferred

only use attribute
names that could also
be variables names

Attribute Names/Values are like Keys/Values

USING DICT

```
d = dict()
```

```
d["x"] = 3  
d["y"] = 4
```

```
tot = d["x"] + d["y"]
```

```
has_z = "z" in d
```

USING

```
class Point:  
    pass
```

```
p = Point()
```

```
setattr(p, "x", 3)  
setattr(p, "y", 4)
```

```
tot = (getattr(p, "x")  
       + getattr(p, "y"))
```

```
has_z = hasattr(p, "z")
```

avoid this

```
p = Point()
```

```
p.x = 3  
p.y = 4
```

```
tot = p.x + p.y
```

```
# no equivalent
```

preferred

only use attribute
names that could also
be variables names

Coding Examples: Animal Classes

Principals

- methods
- checking object type
- type-based dispatch
- self
- constructors

