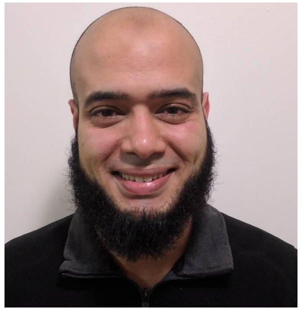
Python Programming
Mutable Objects

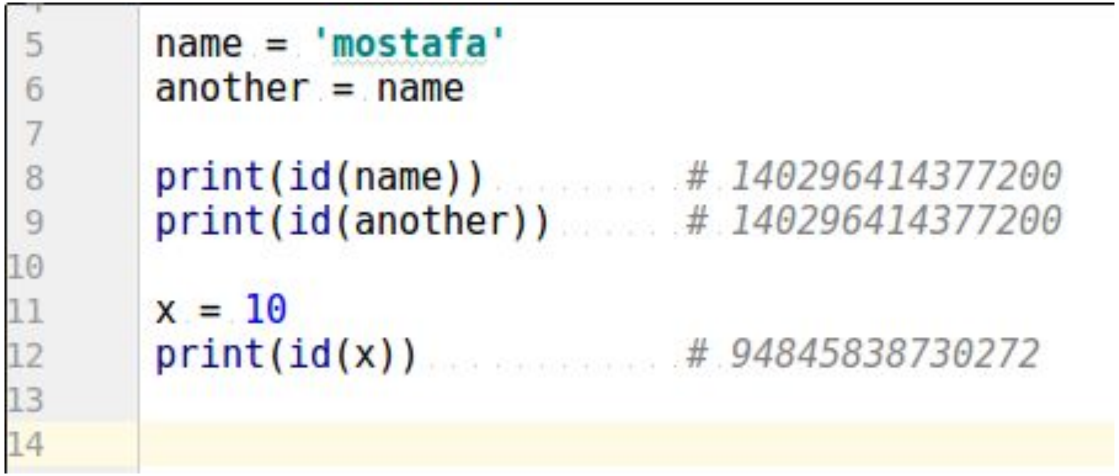
Mostafa S. Ibrahim
Teaching, Training and Coaching since more than a decade!

Artificial Intelligence & Computer Vision Researcher
PhD from Simon Fraser University - Canada
Bachelor / Msc from Cairo University - Egypt
Ex-(Software Engineer / ICPC World Finalist)



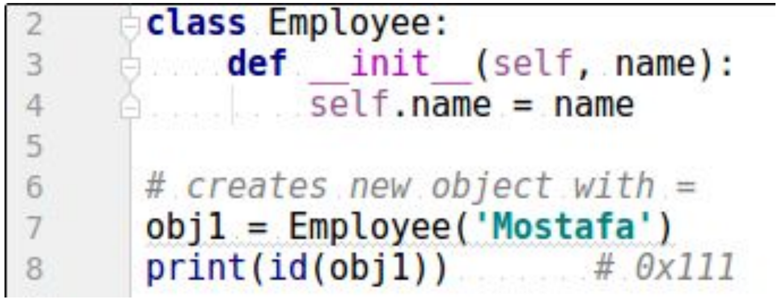
id() function

- id() function returns a unique id for the specified object.
  - In **CPython** impl = It is the object memory address
  - CPython is the reference implementation of the Python programming language.
  - Written in C and Python, CPython is the default and **most widely used** implementation of the language.



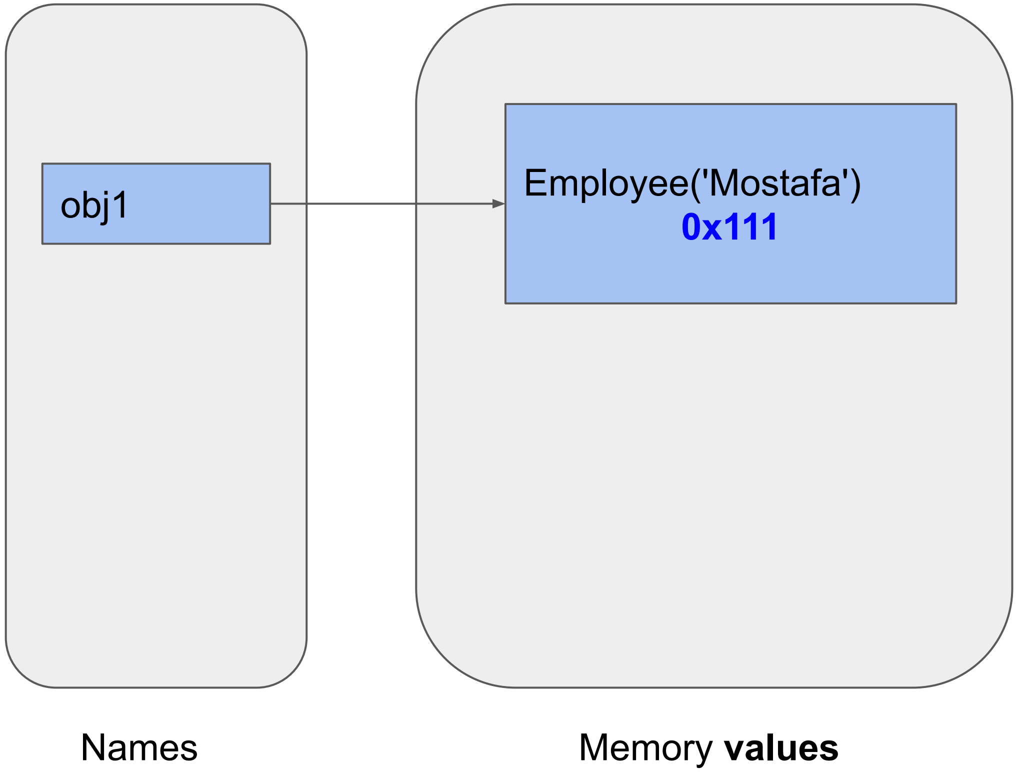
```
name = 'mostafa'
another = name

print(id(name))        # 140296414377200
print(id(another))     # 140296414377200

x = 10
print(id(x))           # 94845838730272

```

The memory

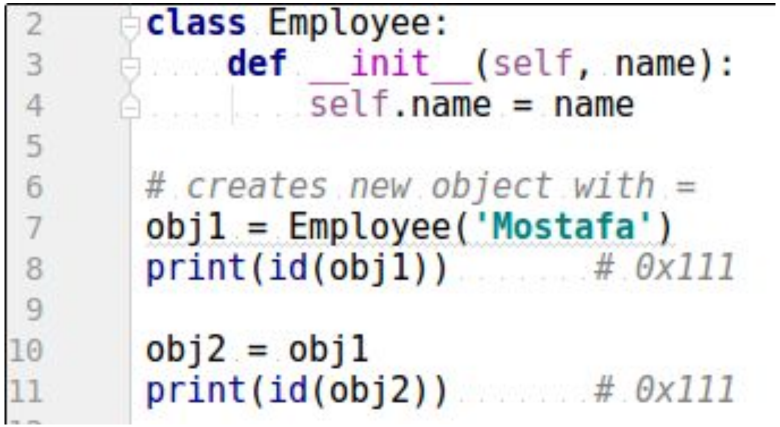


```python
class Employee:
    def __init__(self, name):
        self.name = name

# creates new object with =
obj1 = Employee('Mostafa')
print(id(obj1))        # 0x111
```

- **Names**(Variables) don’t hold **values**
  - They hold **binding** to the value
- **Many** names can refer to **one** value.

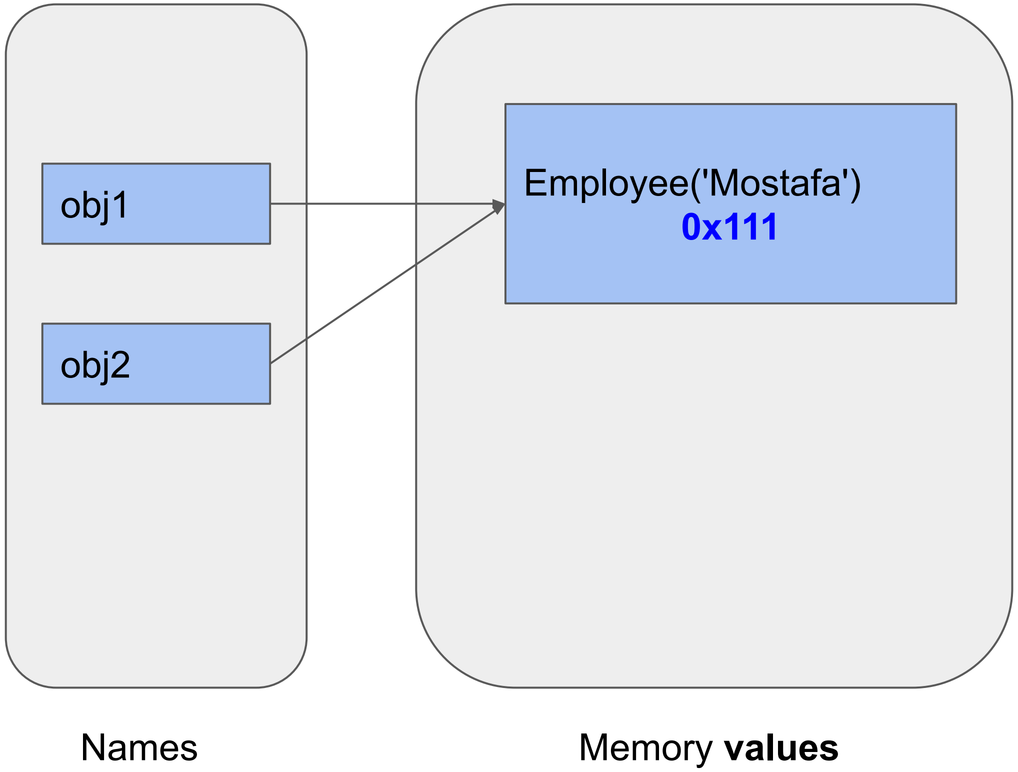


The memory

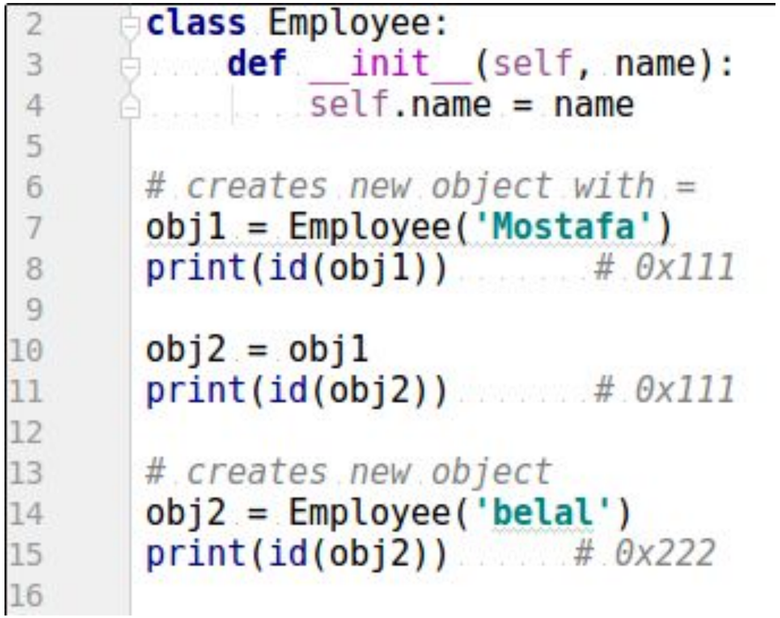


```
class Employee:
    def __init__(self, name):
        self.name = name

# creates new object with =
obj1 = Employee('Mostafa')
print(id(obj1))        # 0x111

obj2 = obj1
print(id(obj2))        # 0x111
```

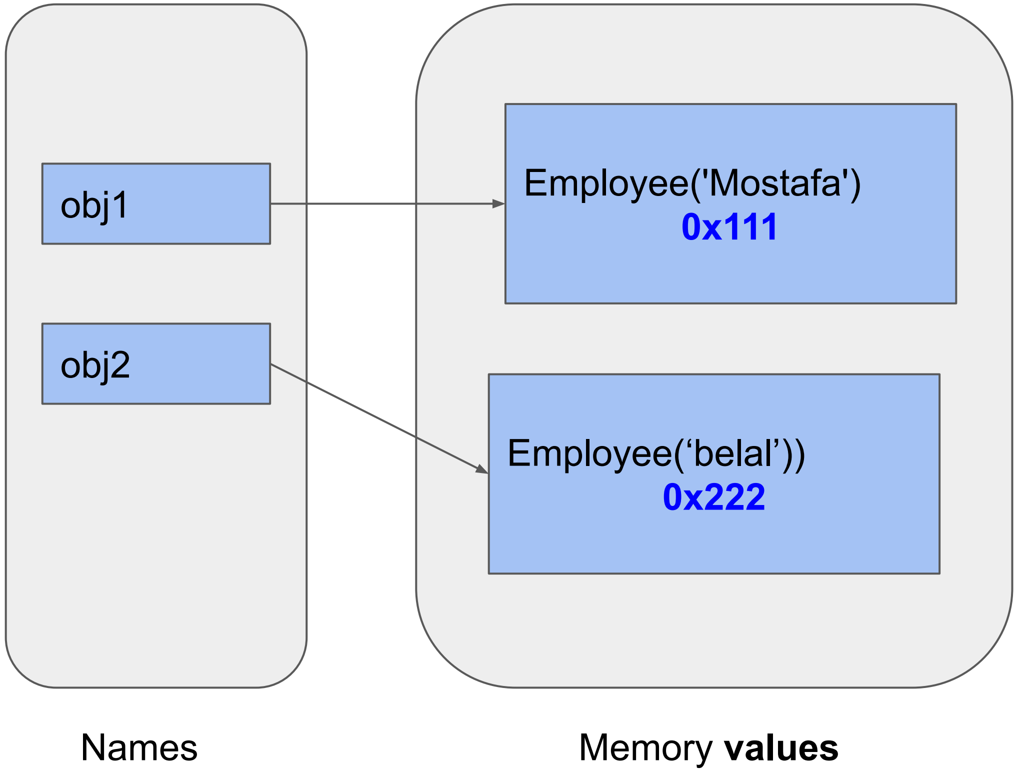
- 2 names: obj1, obj2
- 1 memory object @ 0x111
    - Type: Employee



The memory

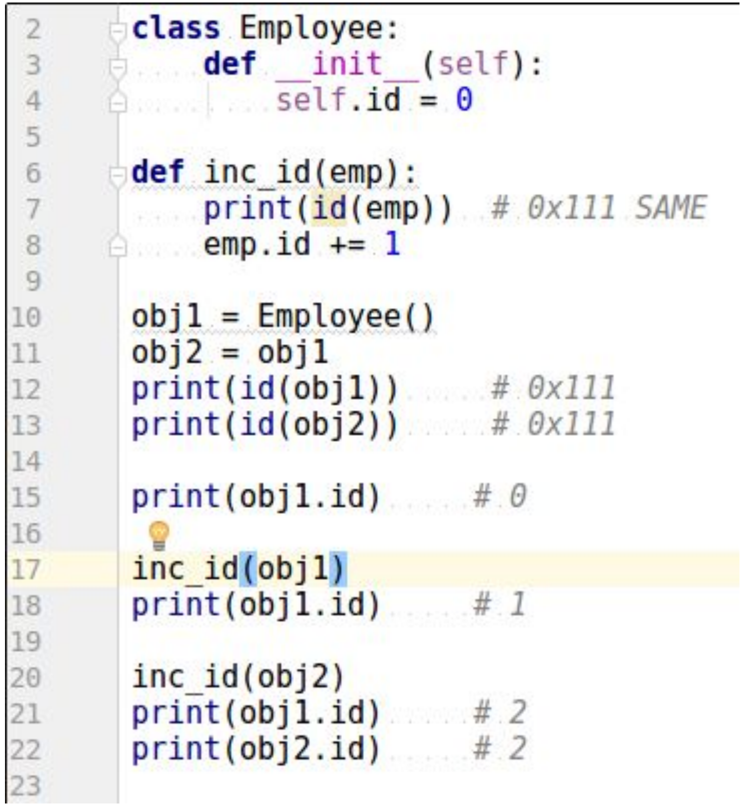


```python
class Employee:
    def __init__(self, name):
        self.name = name

# creates new object with =
obj1 = Employee('Mostafa')
print(id(obj1))         # 0x111

obj2 = obj1
print(id(obj2))         # 0x111

# creates new object
obj2 = Employee('belal')
print(id(obj2))        # 0x222
```



Alias

- 3 names: obj1, obj2, emp
- All of them are bounded to the SAME value
- Any change in one of them is reflected in others



```python
class Employee:
    def __init__(self):
        self.id = 0

def inc_id(emp):
    print(id(emp))  # 0x111 SAME
    emp.id += 1

obj1 = Employee()
obj2 = obj1
print(id(obj1))     # 0x111
print(id(obj2))     # 0x111

print(obj1.id)      # 0

inc_id(obj1)
print(obj1.id)      # 1

inc_id(obj2)
print(obj1.id)      # 2
print(obj2.id)      # 2
```

Mutable Objects

- We created obj1 object
- We also changed its internal values
- Such objects are called **mutable** objects
  - Their value can be changed (in-place)
- So far we studied builtin immutable objects (such as string and int)
  - You can’t change their values! (next lesson)
- Python has built-in mutable classes:
  - list, dict, set, bytearray
- Nice python visualization site: http://pythontutor.com/
- Optional Reading

“Acquire knowledge and impart it to the people.”

“Seek knowledge from the Cradle to the Grave.”