Python Refresher

Variables

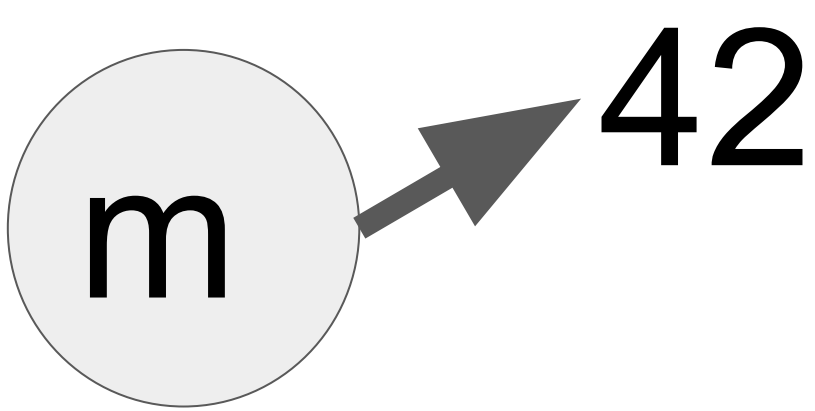
Variables are basically names that refers to values or objects

m = 42
q = “Hello”
pi_number = 3.14

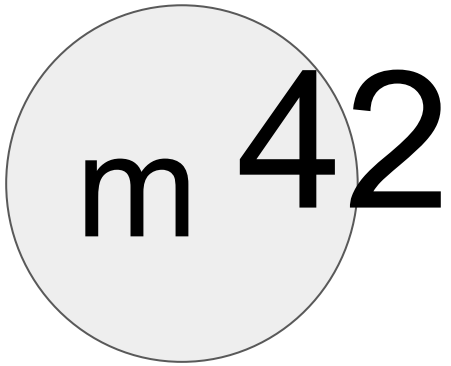
- can be any length
- can consist of uppercase and lowercase letters (A-Z, a-z),
- digits (0-9), t
- the underscore character (_)
- Cannot start with a number

- **One sided:** Assignment operator (=) is not “mathematical equivalence” (one sided)

- **Reference:** Assignment does not copy a value. It attaches a name to the object.



not



Variables, Names, Objects

Strings: Sequences of text characters:

“Hello World”, “a”, “q135”, “34”

Booleans: True or False

Integers: Whole numbers:

23, 493, 20200101

Floats: Numbers with decimal points

3.14159, 55.3

Lists: Holds a group of elements in an ordered way. Can contain zero or more elements. Mutable
[1,10,100]

Dictionaries: Holds a group of elements in an unordered way. Can contain zero or more elements. Mutable
Tuples: Holds a group of elements in an ordered way. Can contain zero or more elements. Immutable
Classes

Indentation in Python

Python uses indentation to define codeblocks/scope:

```
abc = 5

if abc == 10:
   print(“this”)

print(“that”)
```

```
def myfunction():
    print(“this”)
print(“that”)
```

You can use 4 spaces or tab. (4 spaces are the standard).

Table and function based programming (R)

Name	Awards	Partner
Art	10	Paul
Cher	20	Sonny
Paul	20	Art
Sonny	5	Cher

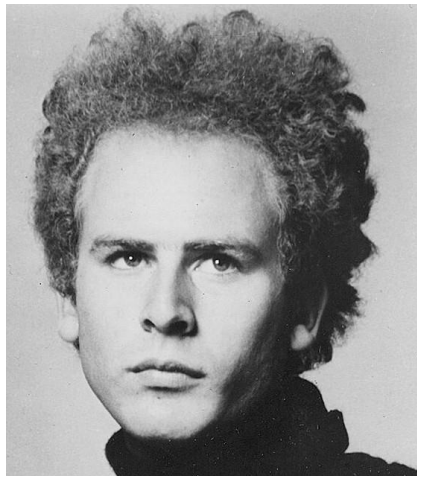
```
function addAwards(name) {
        table[name, ‘awards’] += 10
}
```

```
function MakePartners(name1, name2) {
        table[name1, ‘partner’] = name2
        table[name2, ‘partner’] = name1
}
```

Function-based structure

- Data is stored on a table
- Functions are defined to apply on a table

Object-Oriented Programming



art

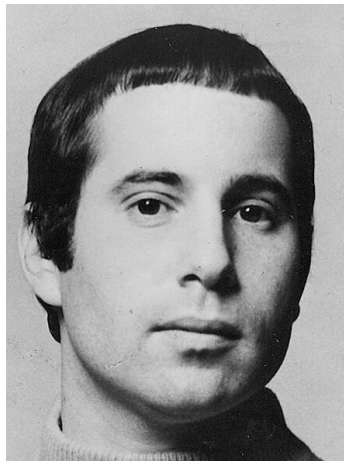
art.name = “art”
art.awards = 10
art.partner = 

art.add_awards()
art.make_partners(person)

cher

cher.name = “cher”
cher.awards = 20
cher.partner = 

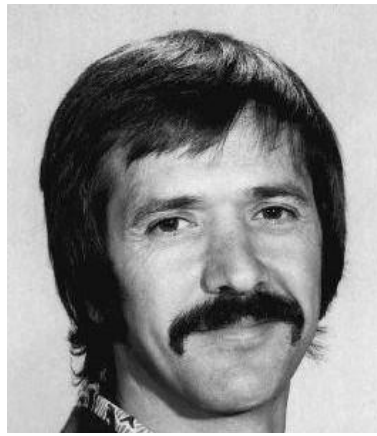
cher.add_awards()
cher.make_partners(person)



paul

paul.name = “paul”
paul.awards = 20
paul.partner = 

paul.add_awards()
paul.make_partners(person)



sonny

sonny.name = “sonny”
sonny.awards = 20
sonny.partner = 

sonny.add_awards()
sonny.make_partners(person)

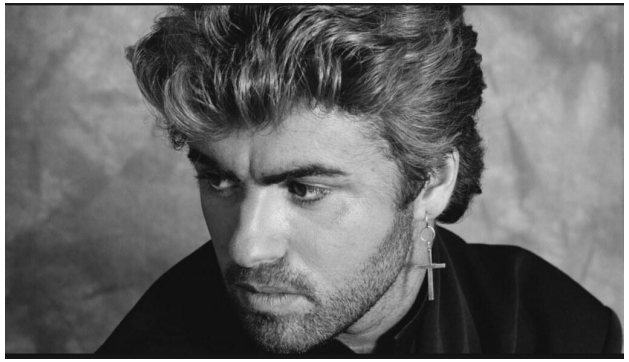
Object-Oriented Programming

Generic Singer: Singer class

Singer

self.name =
self.awards =
self.partner =

self.add_awards()
self.make_partners(person)



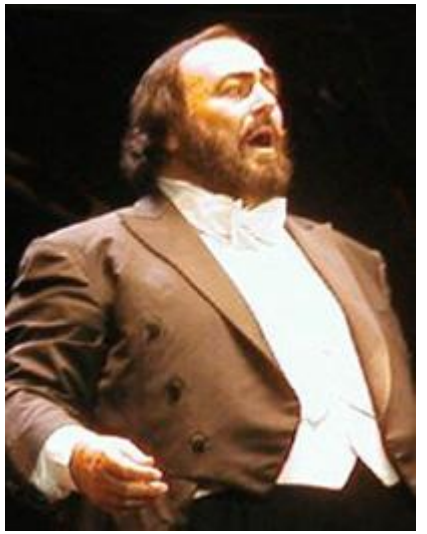
Object-Oriented Programming

Generic Singer: Singer class

Singer

self.name =
self.awards =
self.partner =

self.add_awards()
self.make_partners(person)



pavarotti

vocal_type = “tenor”

preferred_composers = [“verdi”, “puccini”]

callas

vocal_type = “soprano”

preferred_composers = [“donizetti”, “gluck”]

Object-Oriented Programming

Generic Singer: Singer class

Singer

self.name =
self.awards =

self.add_awards()

OperaSinger(Singer)

self.vocal_type =
self.preferred_composers=

PopSinger(Singer)

self.partner =
self.instrument=
self.make_partners(person)

A class in python

- A class is generated with:

```
class MyClass:
    def set_name(self, name):
        self.name = name

    def display_name(self):
        print(self.name)
```

- self refers to instance object being processed
- A “variable” in a class is called “attribute”, a “function” in a class is called “method”
- Calling a class object like a function creates a new instance of it:

```
player1 = MyPlayer()
```

__init__ (Constructor method)

- __init__ method is executed when an instance is created

```
class MyClass:
  def __init__(self, name, points=0):
    self.name = name
    self.points = points

  def display_name(self):
    print(self.name)

player1= MyClass("Joe", 100)
player1.display_name()                # Attributre
        "Joe"
print(player1.points)
        1000
```

- self refers to instance object being processed
- A "variable" in a class is called "attribute", a "function" in a class is called "method"
- Calling a class object like a function creates a new instance of it:

```
player1 = MyPlayer()
```

Classes

Calling a method

- player1.say_hi()

Calling an attribute

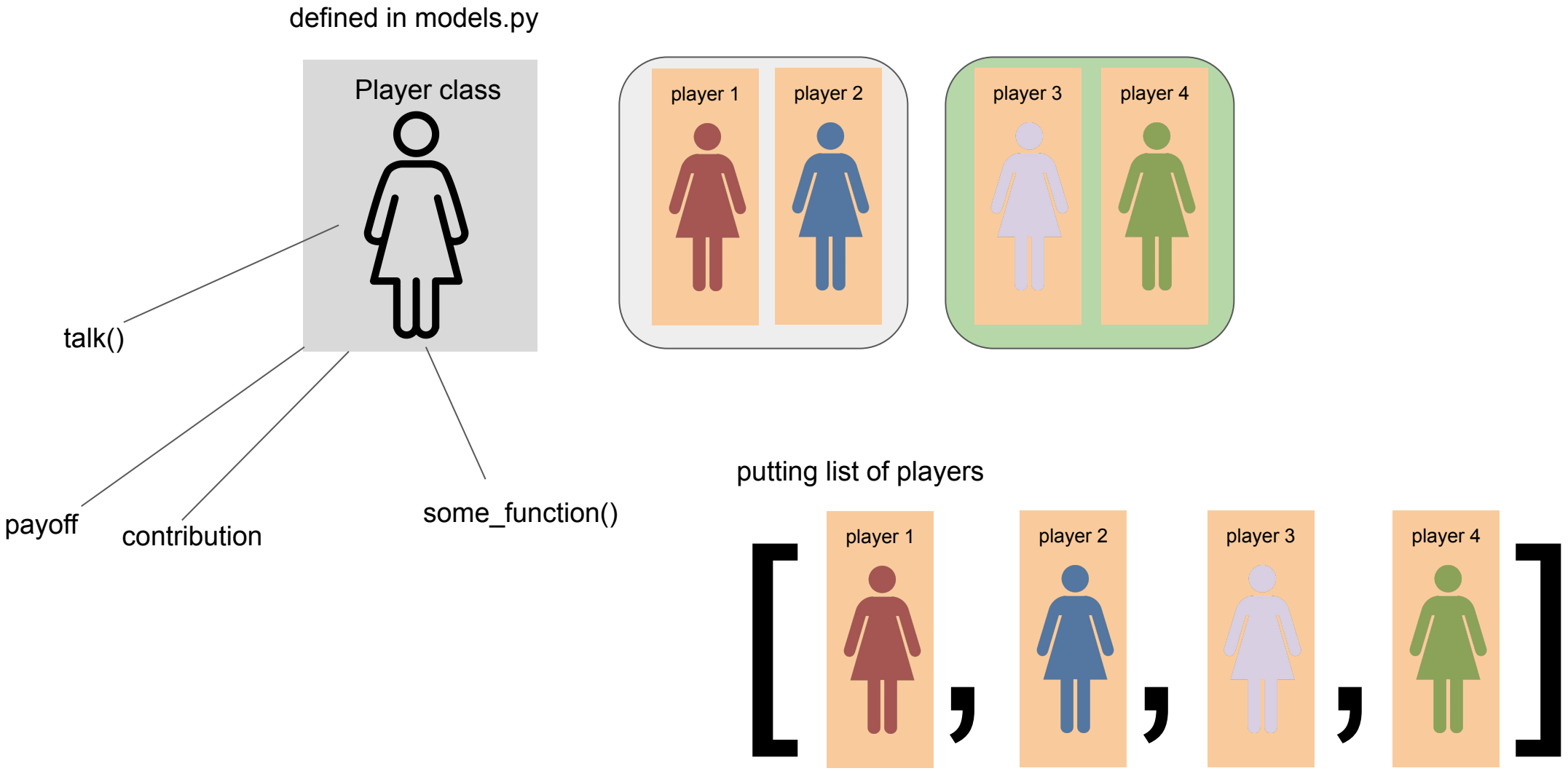
- player1.name

Why use classes?

- Reuse
- Inheritance
- Customization
- Operator Overloading

Class and and instance

A class is a code blueprint for objects. Objects have variables (attributes) and functions (method)



Object-Oriented Programming