The Python Language (Part 4)

Robert M. Dondero, Ph.D.
Princeton University

Objectives

- We will cover:
  - A subset of Python...
  - That is appropriate for COS 333...
  - Through example programs

Agenda

- **Operator overloading**
- Object identity and equality
- Inheritance

Operator Overloading

- See **fraction.py**, **fractionclient.py**

```
$ python fractionclient.py
Numerator 1: 1
Denominator 1: 2
Numerator 2: 3
Denominator 2: 4
frac1: 1/2
frac2: 3/4
frac1 hashcode: -3550055125485641917
frac1 does not equal frac2
frac1 is less than frac2
frac1 is less than or equal to frac2
-frac1: -1/2
frac1 + frac2: 5/4
frac1 - frac2: -1/4
frac1 * frac2: 3/8
frac1 / frac2: 2/3
$
```

Operator Overloading

Special Method	Equivalent
`x.__neg__()`	`-x`
`x.__pos__()`	`+x`
`x.__add__(y)`	`x+y`
`x.__mod__(y)`	`x%y`
`x.__mul__(y)`	`x*y`
`x.__sub__(y)`	`x-y`
`x.__truediv__(y)`	`x/y`
`x.__floordiv__(y)`	`x//y`
`x.__pow__(y)`	`x**y`

Operator Overloading

Special Method	Equivalent
`x.__invert__()`	`~x`
`x.__and__(y)`	`x&y`
`x.__lshift__(y)`	`x<<y`
`x.__or__(y)`	`x|y`
`x.__rshift__(y)`	`x>>y`
`x.__xor__(y)`	`x^y`
`x.__float__()`	`float(x)`
`x.__int__()`	`int(x)`
`x.__str__()`	`str(x)`
`x.__hash__()`	`hash(x)`
`x.__abs__()`	`abs(x)`

Operator Overloading

Special Method	Equivalent
`x.__iadd__(y)`	`x+=y`
`x.__ifloordiv__(y)`	`x//=y`
`x.__imod__(y)`	`x%=y`
`x.__imul__(y)`	`x*=y`
`x.__isub__(y)`	`x-=y`
`x.__itruediv__(y)`	`x//=y`
`x.__iand__(y)`	`x&=y`
`x.__ilshift__(y)`	`x<<=y`
`x.__ior__(y)`	`x|=y`
`x.__irshift__(y)`	`x>>=y`
`x.__ixor__(y)`	`x^=y`
`x.__ipow__(y)`	`x**=y`

Operator Overloading

Special Method	Equivalent
`x.__getitem__(y)`	`x[y]`
`x.__setitem__(y, z)`	`x[y] = z`
`x.__contains__(y)`	`y in x`
`x.__delitem__(y)`	`del(x[y])`
`x.__len__()`	`len(x)`

And many more!

Agenda

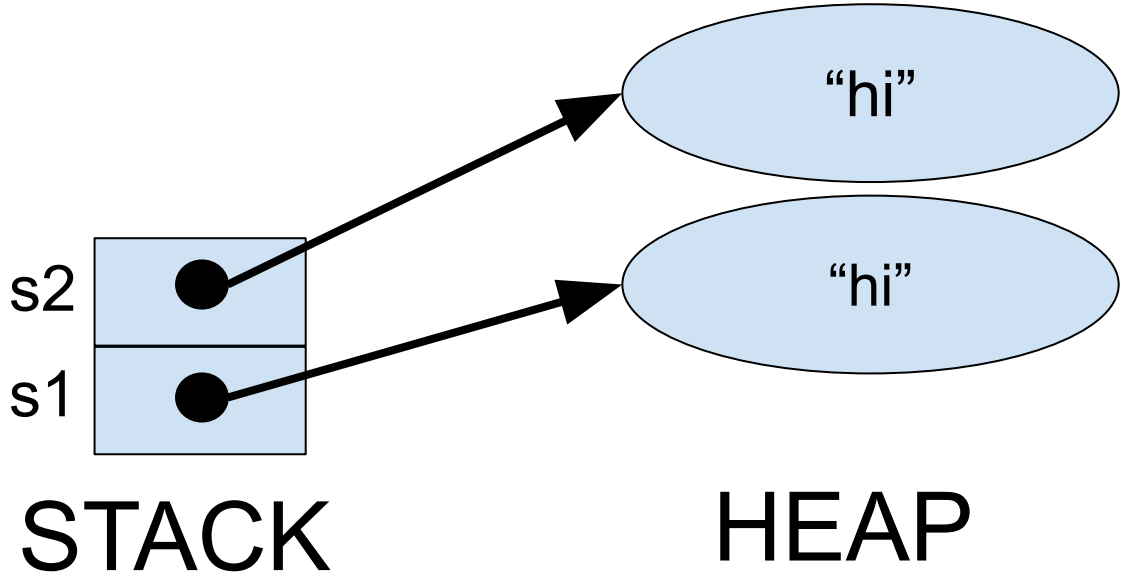
- Operator overloading
- **Object identity and equality**
- Inheritance

Object Identity and Equality

- See **EqualityStr.java**

```
$ javac EqualityStr.java
$ java EqualityStr
Enter the first string:
hi
Enter the second string:
hi
false
true
$
```

Object Identity and Equality



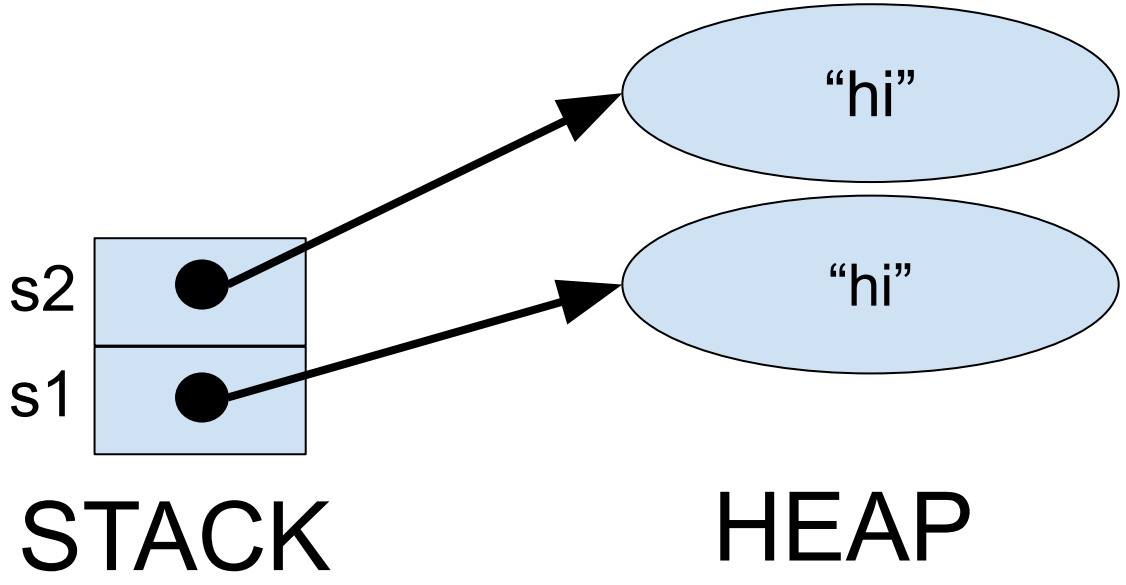

```
s1 == s2 => false
s1.equals(s2) => true
```

Object Identity and Equality

- See **equalitystr.py**

```
$ python equalitystr.py
Enter the first string:
hi
Enter the second string:
hi
False
True
$
```

Object Identity and Equality




```
s1 is s2 => False
s1 == s2 => s1.__eq__(s2) => True
```

Object Identity and Equality

Java, for type String:

Expression	Compares
`s1 == s2`	Object references
`s1.equals(s2)`	Objects

Python, for type str:

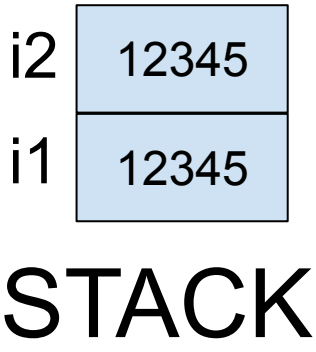
Expression	Compares
`s1 is s2`	Object references
`s1 == s2` `s1.__eq__(s2)`	Objects

Object Identity and Equality

- See **EqualityInt.java**

```
$ javac EqualityInt.java
$ java EqualityInt
Enter the first int:
12345
Enter the second int:
12345
true
$
```

Object Identity and Equality




```
i1 == i2 => true
```

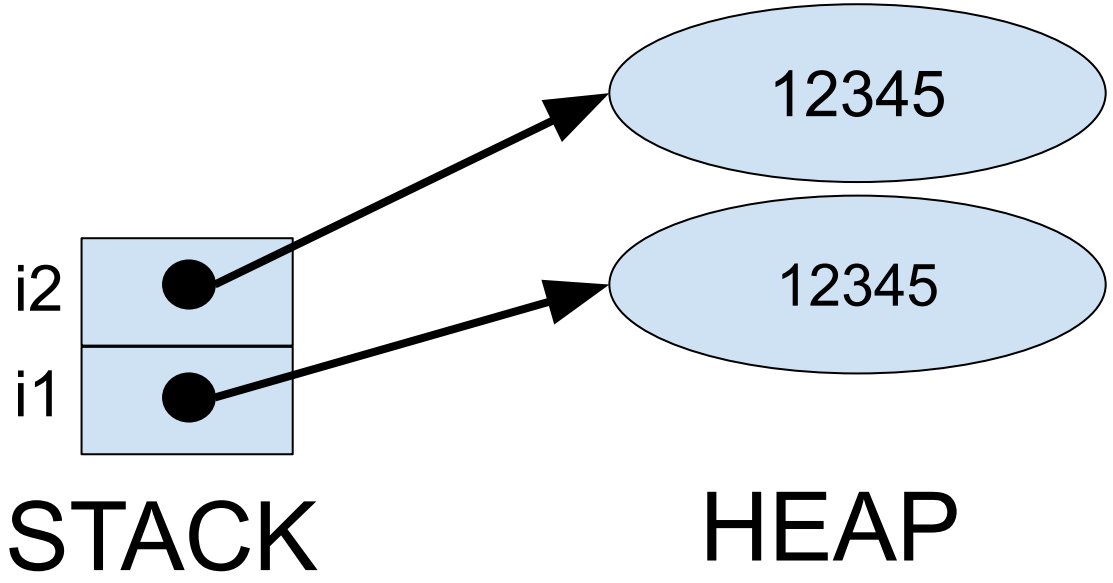
~~`i1.equals(i2) => Illegal`~~

Object Identity and Equality

- See **equalityint.py**

```
$ python equalityint.py
Enter the first int:
12345
Enter the second int:
12345
False
True
$
```

Object Identity and Equality




```
i1 is i2 => False
i1 == i2 => i1.__eq__(i2) => True
```

Object Identity and Equality

Java, for type int:

Expression	Compares
`i1 == i2`	Values
~~`i1.equals(i2)`~~	Illegal

Python, for type int:

Expression	Compares
`i1 is i2`	Object references
`i1 == i2` `i1.__eq__(i2)`	Objects

Object Identity and Equality

Language	Characteristics
Java	**Hybrid** OO language
Python	**Pure** OO language

Java:

Expression	Compares
`x == y`	(sometimes) Values (sometimes) Object references
`x.equals(y)`	Objects

Python:

Expression	Compares
`x is y`	Object references
`x == y` `x.__eq__(y)`	Objects

Agenda

- Operator overloading
- Object identity and equality
- **Inheritance**

Inheritance

- See **queue.py**, **priorityqueue.py**, **priorityqueueclient.py**

```
$ python priorityqueueclient.py
Enter non-negative ints, one per line.
Enter a negative int to stop.
4
8
5
6
-1
8
6
5
4
$
```

Summary

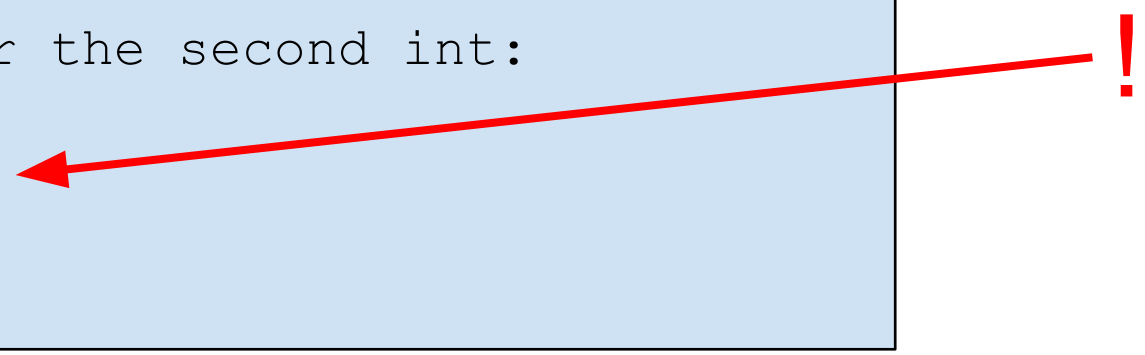
- We have covered these aspects of Python:
  - Operator overloading
  - Object identity and equality
  - Inheritance
- See also:
  - **Appendix 1**: Predefined Objects
  - **Appendix 2**: Objects as Arguments and Parameters

Appendix 1: Predefined Objects

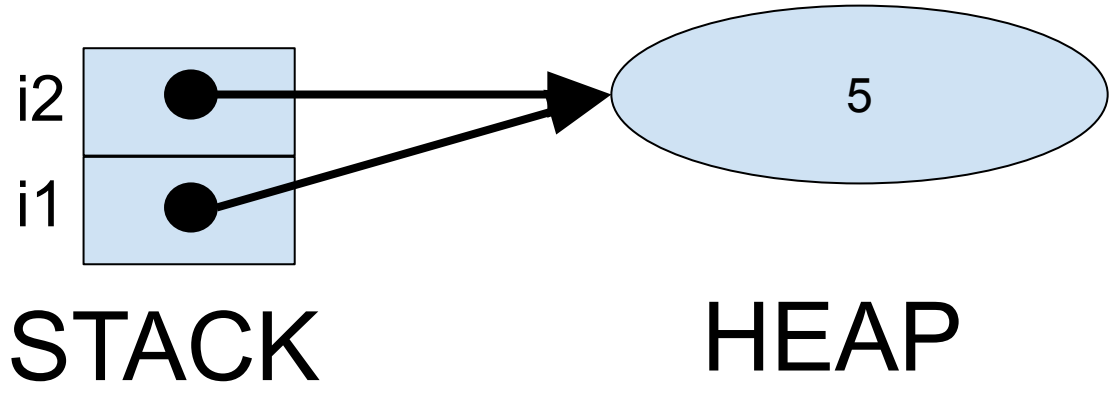
Predefined Objects

- Recall equalityint.py

```
$ python equalityint.py
Enter the first int:
5
Enter the second int:
5
True
True
$
```



Predefined Objects




```
i1 is i2 => True
i1 == i2 => i1.__eq__(i2) => True
```

Predefined Objects

- Explanation:
  - For efficiency, cPython creates `int` objects -5…256 at process startup
    - At any time:
      - Only one -5 `int` object exists
      - Only one -4 `int` object exists
      - …
      - Only one 256 `int` object exists

Predefined Objects

- And, incidentally…
  - For efficiency, cPython creates `bool` objects `True` and `False` at process startup
    - At any time:
      - Only one `True` object exists
      - Only one `False` object exists

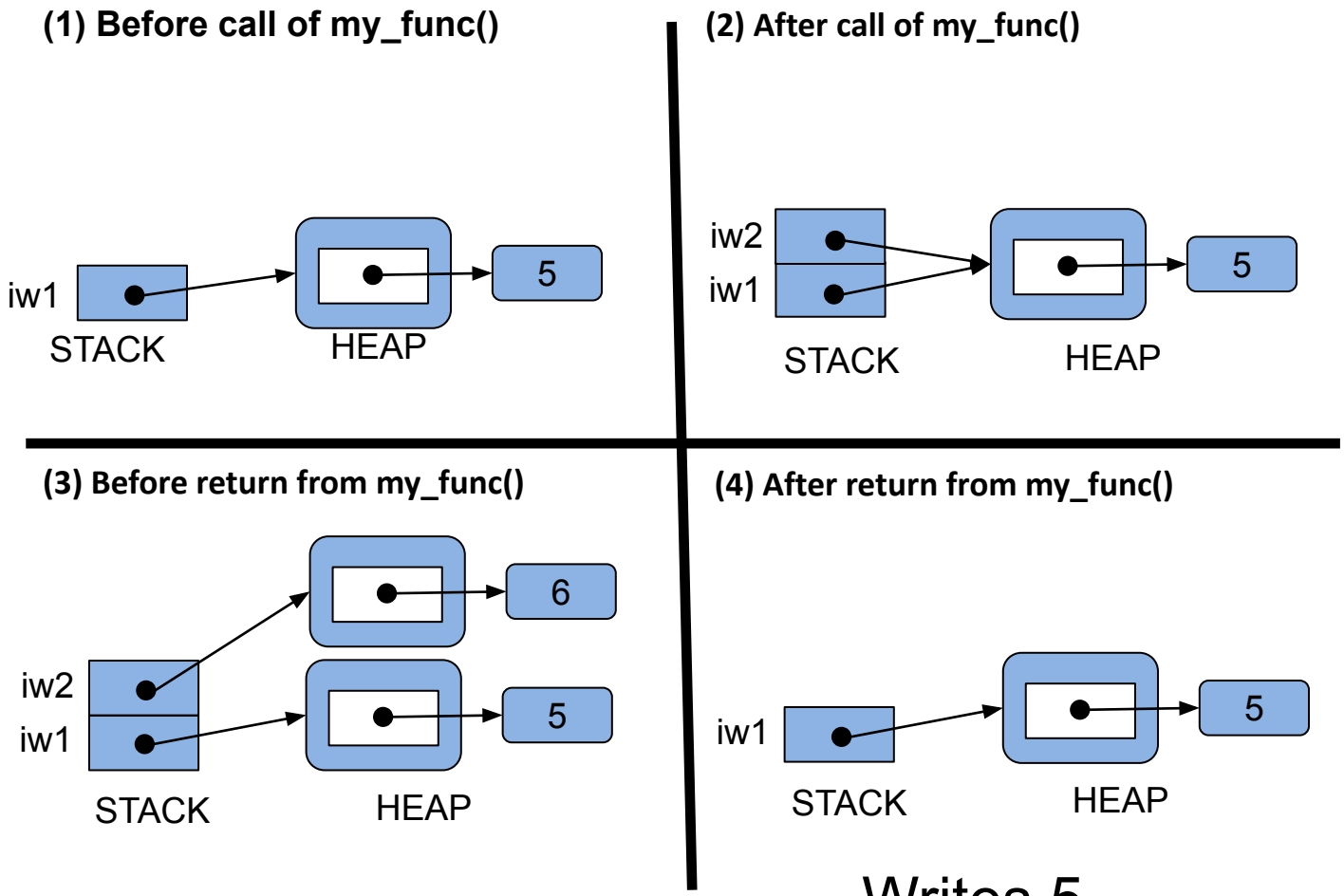
Appendix 2: Objects as Arguments and Parameters

Objects as Args and Params

- See **objectparam1.py**
  - What does it write?

Objects as Args and Params

See **objectparam1.py** (cont.)



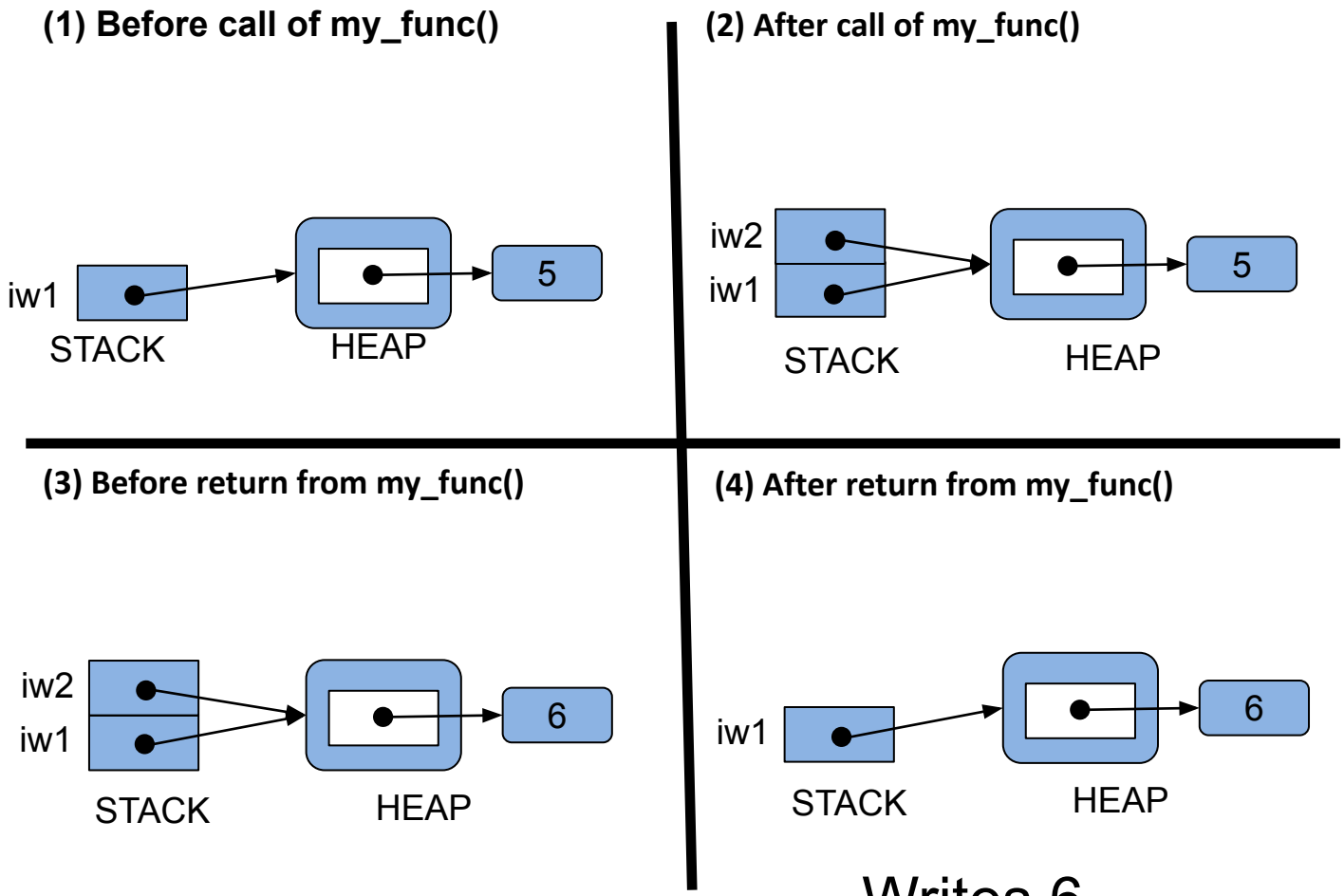
Writes 5

Objects as Args and Params

- See **objectparam2.py**
  - What does it write?

Objects as Args and Params

See **objectparam2.py** (cont.)



Writes 6

Objects as Args and Params

- In Python
  - As in Java…
  - Objects are passed by reference
  - More precisely…
  - Object references are passed by value