COMP110: Principles of Computing

10: References

Research journal

Research journal

- **Read** some seminal papers in computing (listed on the assignment brief)
- **Choose** one of them
- **Research** how this paper has influenced the field of computing
- **Write up** your findings
  - Maximum 1500 words
  - With reference to appropriate academic sources

Marking rubric

See assignment brief on LearningSpace/GitHub

Timeline

- **Peer review** next week! (4th December)
- **Deadline** shortly after! (check MyFalmouth)

Pass by reference

References

- Our picture of a variable: a labelled box containing a value
- For "plain old data" (e.g. numbers), this is accurate
- For **objects** (i.e. instances of classes), variables actually hold **references** (a.k.a. **pointers**)
- It is possible (indeed common) to have **multiple references** to the same underlying object

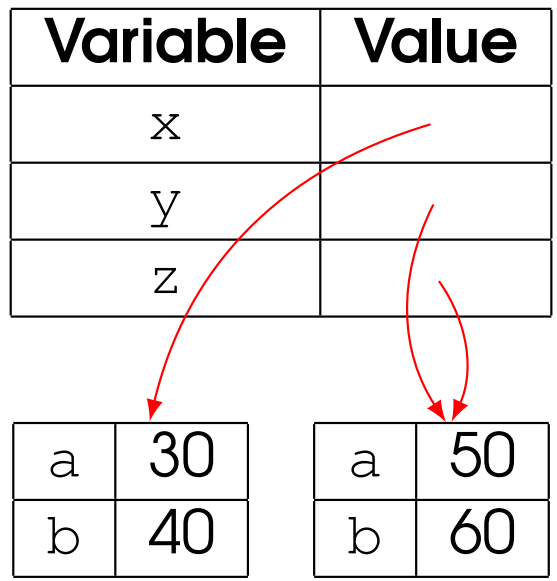
The wrong picture

```
class Thing:
    def __init__(self,
                 a, b):
        self.a = a
        self.b = b

x = Thing(30, 40)
y = Thing(50, 60)
z = y
```

Variable	Value	
x	a	30
	b	40
y	a	50
	b	60
z	a	50
	b	60

The right picture

```
class Thing:
    def __init__(self,
                 a, b):
        self.a = a
        self.b = b

x = Thing(30, 40)
y = Thing(50, 60)
z = y
```



Values and references

Socrative room code: `FALCOMPED`

```
a = 10
b = a
a = 20
print("a:", a)
print("b:", b)
```

Values and references

Socrative room code: FALCOMPED

```
class X:
    def __init__(self, value):
        self.value = value

a = X(10)
b = a
a.value = 20
print("a:", a.value)
print("b:", b.value)
```

Values and references

Socrative room code: FALCOMPED

```
class X:
    def __init__(self, value):
        self.value = value

a = X(10)
b = X(10)
a.value = 20
print("a:", a.value)
print("b:", b.value)
```

Pass by value

In **function parameters**, "plain old data" is passed by **value**

```
def double(x):
    x *= 2

a = 7
double(a)
print(a)
```

`double` does not actually do anything, as `x` is just a local copy of whatever is passed in!

Pass by reference

However, instances are passed by **reference**

```
class Box:
    def __init__(self, v):
        self.value = v

def double(x):
    x.value *= 2

a = Box(7)
double(a)
print(a.value)
```

`double` now has an effect, as `x` gets a reference to the `Box` instance

Lists are objects too

```
a = ["Hello"]
b = a
b.append("world")
print(a)  # ["Hello", "world"]
```

... which means you should be careful when passing lists into functions, because the function might actually change the list!

References can be circular

```
class X:
    pass

foo = X()
foo.x = foo
foo.y = "Hello"

print(foo.x.x.x.x.x.y)
```

References and pointers

- Some languages (e.g. C, C++) use **pointers**
- Pointers are a type of reference, and have the same semantics
- C++ also has something called references...