CSC108: Introduction to Computer Programming

Lecture 5

Wael Aboulsaadat

Acknowledgment: these slides are based on material by: Velian Pandeliev, Diane Horton, Michael Samozi, Jennifer Campbell, and Paul Gries from CS UoT

Announcements

- Midterm coming week
- Samples will be posted

What have we learnt up till now?

- Variables
- Logical & Mathematical Operators
- Assignment Statement
- Types & Type conversion
- if/else Statement
- print
- input & raw_input
- Functions
- Docstrings
- while loops
- Variable scope & Namespaces

Object Oriented Programming

Object oriented paradigm

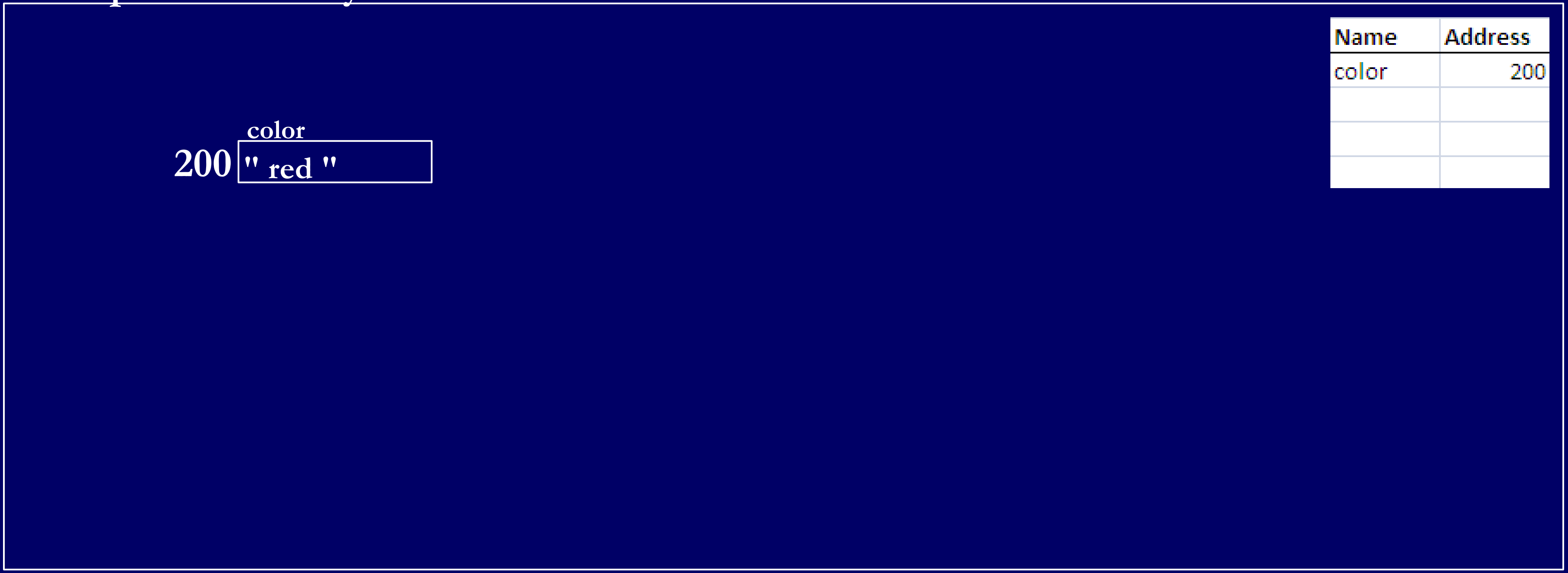
- Modeling real world object
  - Form + function in one enclosure
- Blueprint vs. physical object
- Benefits:
  - Encapsulation of data + behavior in one entity
  - Can model real life objects in our programs

Computer memory

- Every memory cell has an address

```
>>> color =“red”
```

Computer Memory



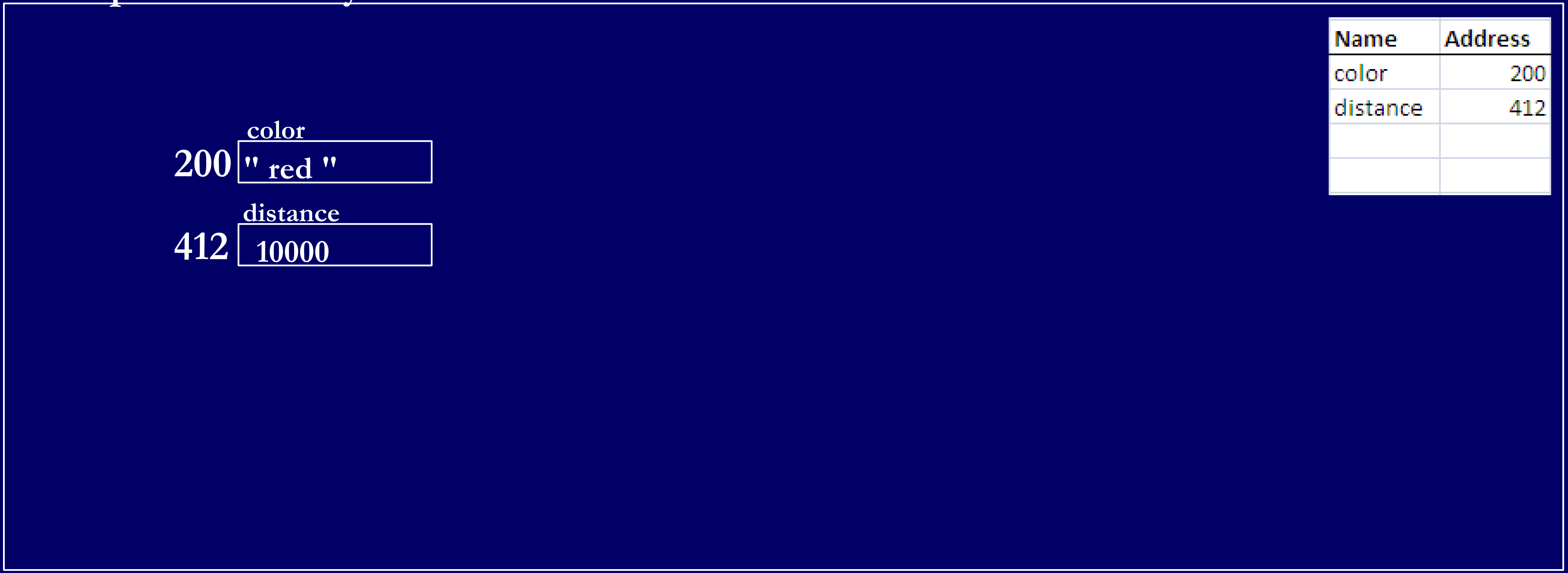
Name	Address
color	200

Computer memory

- Every memory cell has an address

```
>>> color =“red”
>>> distance = 1000
```

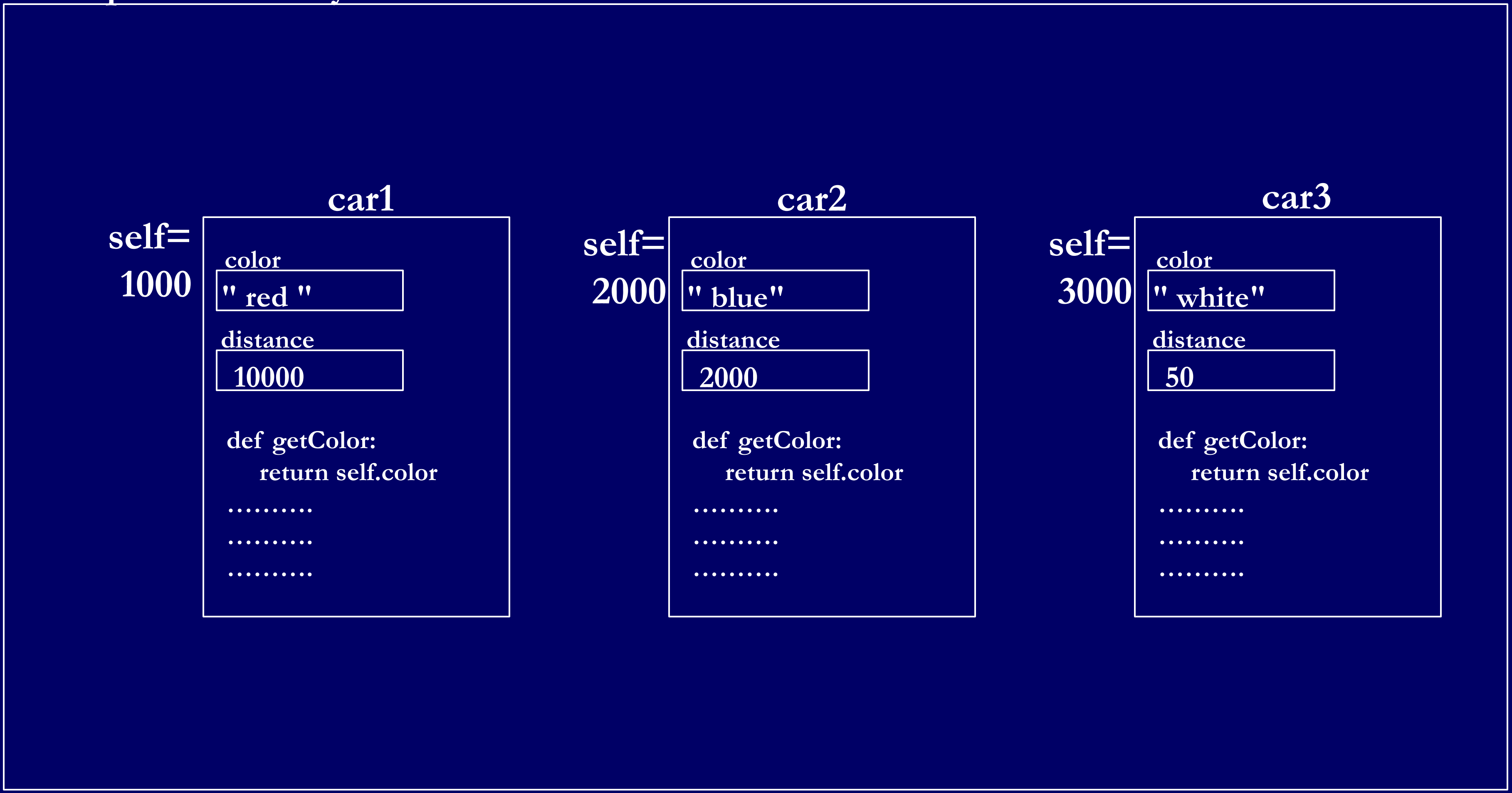
Computer Memory



Name	Address
color	200
distance	412

Class car example

Computer Memory



Lists (revisited)

List Functions

- *item in listname - evaluates to True iff item* is found in *listname*

- *item not in listname - evaluates to True iff item is not found in listname*

```
>>> a = [1,2,3]
>>> 2 in a
True
>>> 5 in a
False
```

List Functions

- len(a) - returns the number of items in list a
- max(a) - returns the largest element in list a
- min(a) - returns the smallest element in list a
- sum(a) - returns the sum of all elements in list a

List Methods

- lst.index(item) - returns the index of the first time item appears in lst
- lst.count(item) - returns the number of times item appears in lst
- lst.insert(index, item) - inserts item before position index in lst
- lst.remove(item) - removes the **first** occurrence of item from lst

List Methods II

- lst.append(item) - adds item to the end of lst
- lst.pop(index) - removes and returns the item at index. If index is not supplied, removes and returns the last item in lst
- lst.extend(anotherlist) - adds all the items in anotherlist to the end of lst
- lst.reverse() - reverses the order of items in lst
- lst.sort() - sorts the items in lst in alphanumeric order

Pitfalls with Lists: using loop variable incorrectly

- Say some instructor had very low class marks in a midterm
- We can write a program to boost everyone's marks so that instructor doesn't get in trouble with his/her department.

```
marks = [44,65,72,23,54,39]
for a in marks:
        a = a + 10
```

- Right?

Pitfalls with Lists: using loop variable incorrectly

- Say some instructor had very low class marks in a midterm
- We can write a program to boost everyone's marks so that instructor doesn't get in trouble with his/her department.

```
marks = [44,65,72,23,54,39]
i = 0
for a in marks:
        marks[i] = marks[i] + 10
        i = i + 1
```

- Right? Wrong! Modifying a does not change the list!

Pitfalls with Lists: using loop variable incorrectly

- Say some instructor had very low class marks in a midterm
- We can write a program to boost everyone's marks so that instructor doesn't get in trouble with his/her department.

```
marks = [44,65,72,23,54,39]
i = 0
while i < len(marks):
        marks[i] = marks[i]  + 10
        i =  i + 1
```

Pitfalls with Lists: accessing a list element that does not exist!

- What is the problem here?

```
marks = [44,65,72,23,54,39]
i = 0
while i <= len(marks):
        marks[i] = marks[i]  + 10
        i =  i + 1
```

Strings (revisited)

upper() and lower()

- The first two methods deal with capitalization:

  s.upper() returns a copy of s converted to uppercase.

  s.lower() returns a copy of s converted to lowercase.

- And, if you ever need it:

  s.capitalize() returns a copy of s with the first character converted to uppercase.

find() and replace()

- Some string methods take other parameters:

  s.find(a) returns the index of the first occurrence of substring a in s or -1 if a is not in s

  s.replace(a,b) returns a string with all occurrences of substring a in s replaced by the string b

  s.count(a) returns the number of occurrences of substring a in s

startswith() and split()

- s.startswith(a) returns True iff substring a is at the beginning of s
- s.endswith(a) returns True iff substring a is at the end of s
- s.split(a) returns a list of the word in s using substring a as the delimeter. If a is not supplied, split() uses spaces.

Chaining Methods

- Methods can be chained together:

```
s.lower().count('a')
```

- They are evaluated left to right.

```
>>> s = "Hello"
>>> s.startswith("h")
>>> False
>>> s.lower().startswith("h")
>>> True
```

What have we learnt today?

- Object Oriented Programming
- Lists functions and methods
- Strings functions and methods

This Week’s To Do List

- Go through lecture slides – make sure you try the code snippets
- Try the lecture’s programs posted on course website