Lecture 4

Strings & Objects

Netids That Did Not Do the Quiz

- aal59
- abr75
- ank43
- cms242
- egm58
- gbf22
- gem67
- gj54
- xl237
- hy388
- jbm247
- jtk53
- ksk75
- kt429
- meb327
- mrr87
- srh78

Announcements for this Lecture

Do the Quiz!

- No quiz; cannot take course
- You have one last time!
- Also remember the survey

Readings

- Chapter 8 (not 8.6, 8.11)
- Sections 3.5 – 3.13

Today's Lab

- Similar to last week's lab
  - Still answering a worksheet
  - Not really writing programs
  - You will be using modules, but not writing them
- Preparation for Assignment 1
  - Do not leave the lab before you finish the String section
  - Okay to do the rest at home

[xkcd.com]

String: Text as a Value

- String are quoted characters
  - `'abc d'` (Python prefers)
  - `"abc d"` (most languages)
- How to write quotes in quotes?
  - Delineate with "other quote"
  - **Example**: `" ' "` or `' " '`
  - What if need both `"` and `'` ?
- **Solution**: escape characters
  - Format: \ + letter
  - Special or invisible chars

Type: `str`

Char	Meaning
`\'`	single quote
`\"`	double quote
`\n`	new line
`\t`	tab
`\\`	backslash

String are Indexed

- s = 'abc d'

0	1	2	3	4
a	b	c		d

- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"

- s = 'Hello all'

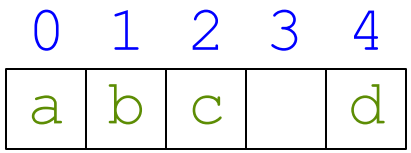
0	1	2	3	4	5	6	7	8
H	e	l	l	o		a	l	l

- What is s[3:6]?

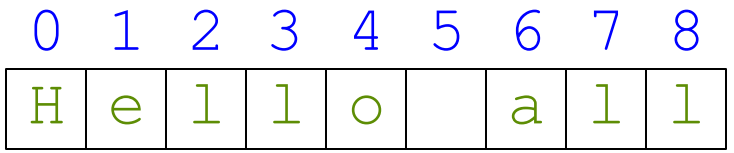
A: 'lo a'
B: 'lo'
C: 'lo '
D: 'o '
E: I do not know

String are Indexed

- s = 'abc d'



- s = 'Hello all'



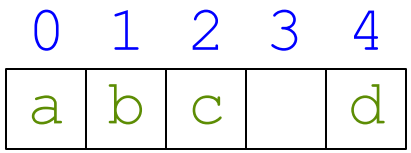
- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"

- What is s[3:6]?

A: 'lo a'
B: 'lo'
C: 'lo ' **CORRECT**
D: 'o '
E: I do not know

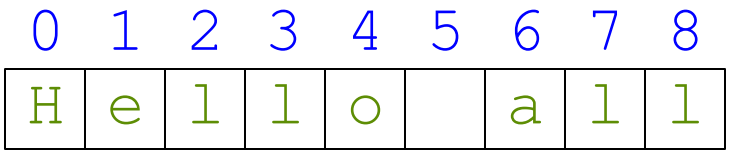
String are Indexed

- s = 'abc d'



- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"

- s = 'Hello all'

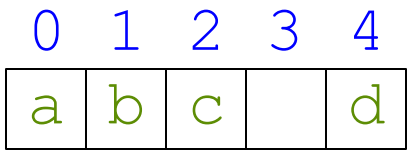


- What is s[:4]?

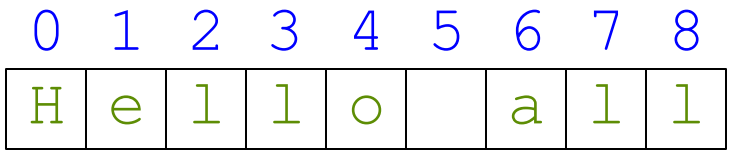
A: 'o all'
B: 'Hello'
C: 'Hell'
D: Error!
E: I do not know

String are Indexed

- s = 'abc d'



- s = 'Hello all'



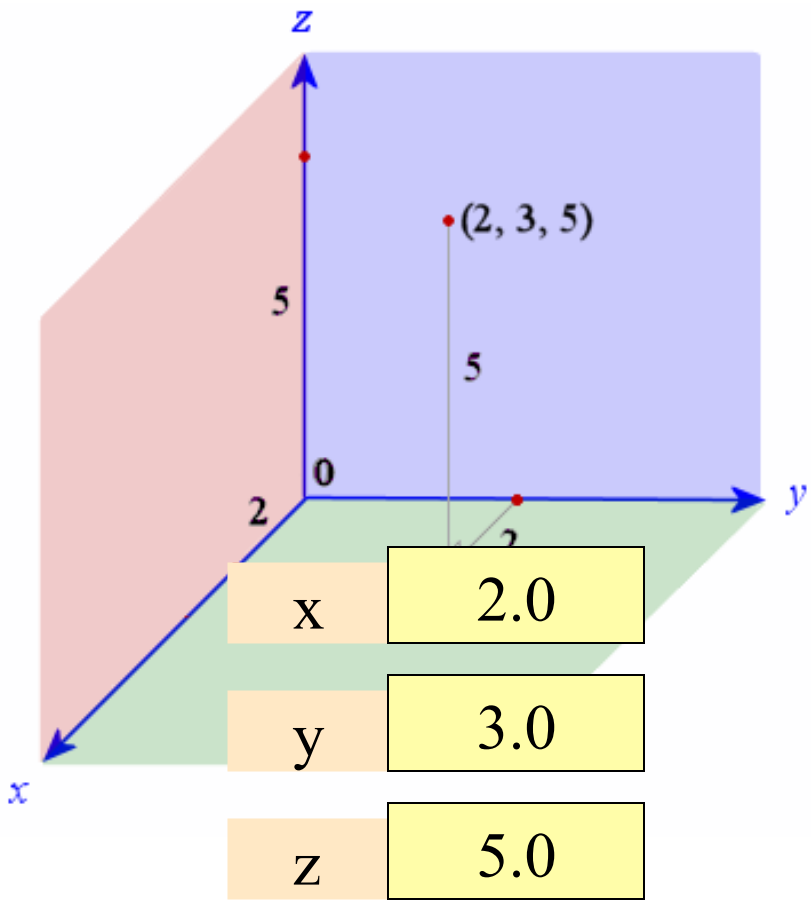
- Access characters with []
  - s[0] is 'a'
  - s[4] is 'd'
  - s[5] causes an error
  - s[0:2] is 'ab' (excludes c)
  - s[2:] is 'c d'
- Called "string slicing"

- What is s[:4]?

A: 'o all'
B: 'Hello'
C: 'Hell' **CORRECT**
D: Error!
E: I do not know

Type: Set of values and the operations on them

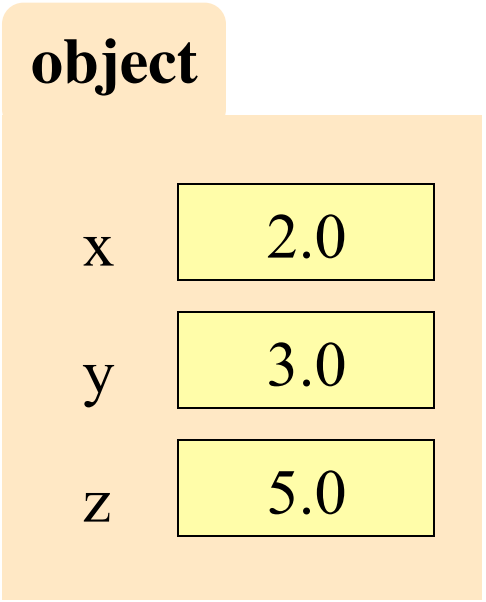
- Want a point in 3D space
  - We need three variables
  - x, y, z coordinates
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy
- We need a new **type**



Type: Set of values and the operations on them

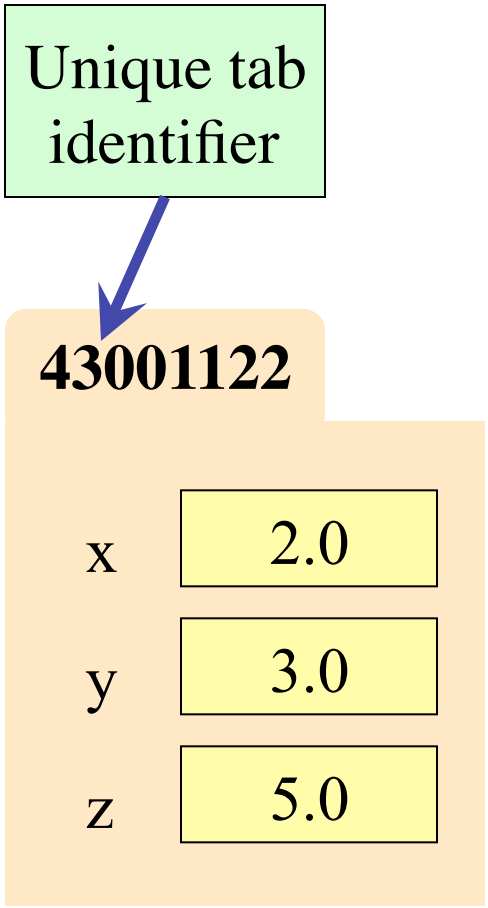
- Want a point in 3D space
  - We need three variables
  - x, y, z coordinates
- What if have a lot of points?
  - Vars x0, y0, z0 for first point
  - Vars x1, y1, z1 for next point
  - …
  - This can get really messy
- We need a new **type**

- Can we stick them together in a "folder"?
- Motivation for **objects**



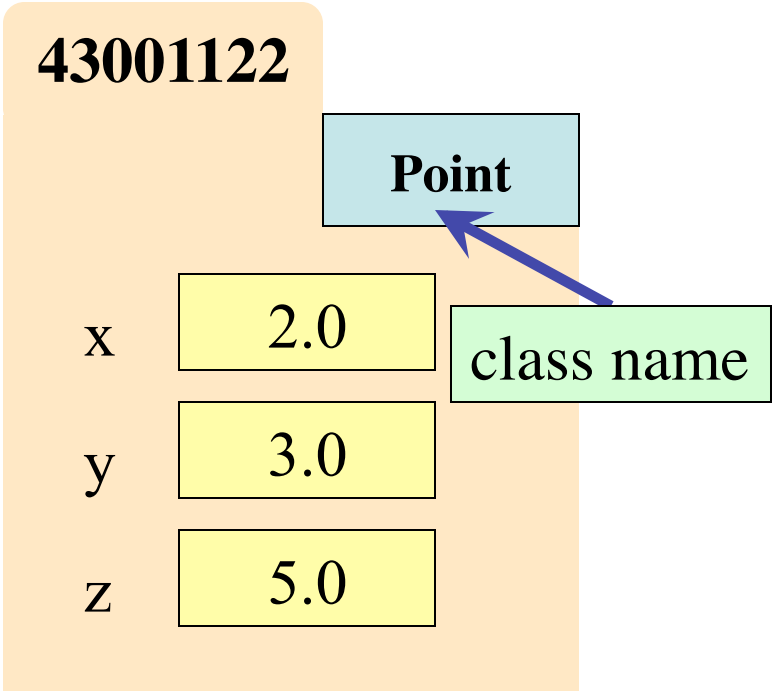
Objects: Organizing Data in Folders

- An object is like a **manila folder**
- It contains other variables
  - Variables are called **attributes**
  - Can change values of an attribute (with assignment statements)
- It has a "tab" that identifies it
  - Unique number assigned by Python
  - You cannot ever change this
  - More on this in demo later



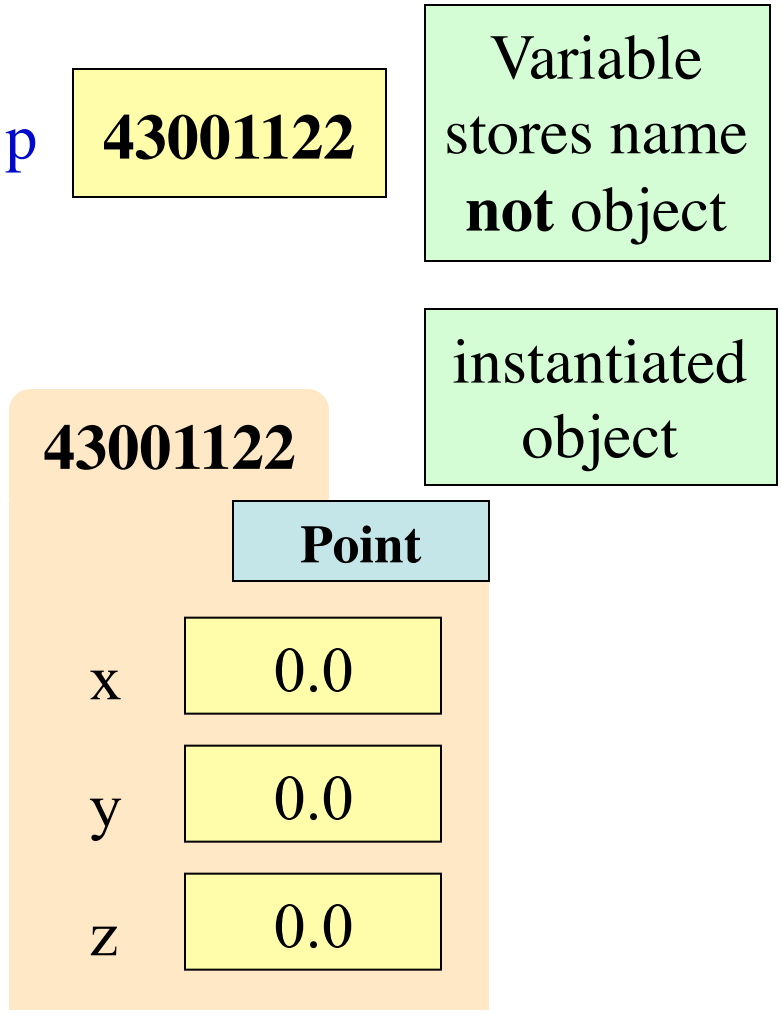
Classes: Types for Objects

- Values must have a type
  - An object is a **value**
  - Object type is a **class**
- **Modules** provide classes
  - **Example**: point.py
  - Import to use Point
- Will cover classes later
  - Do not try to understand the contents of point.py
  - Lot more to learn first



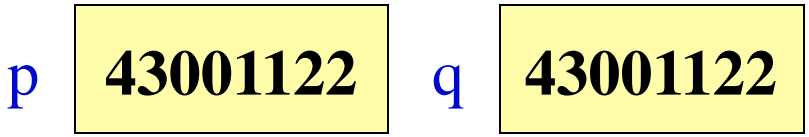
Constructor: Function to make Objects

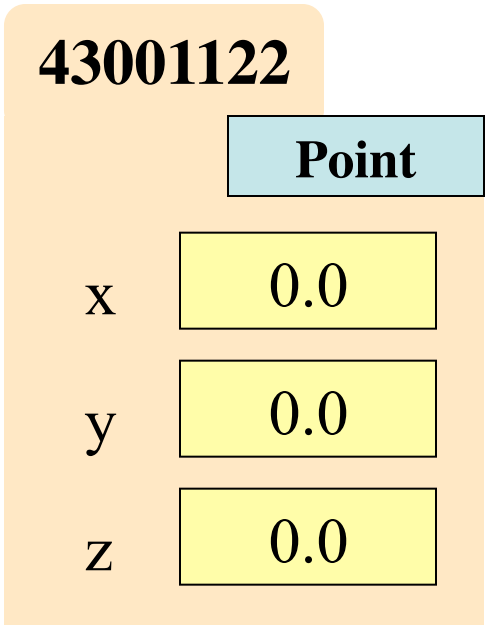
- How do we create objects?
  - Other types have **literals**
  - **Example**: 1, "abc", **true**
  - No such thing for objects
- **Constructor Function**:
  - Same name as the class
  - **Example**: Point(0,0,0)
  - Makes an object (manila folder)
  - Returns folder name as value
- **Example**: p = Point(0,0,0)
  - Creates a Point object
  - Stores value (tab name) in p



Object Variables

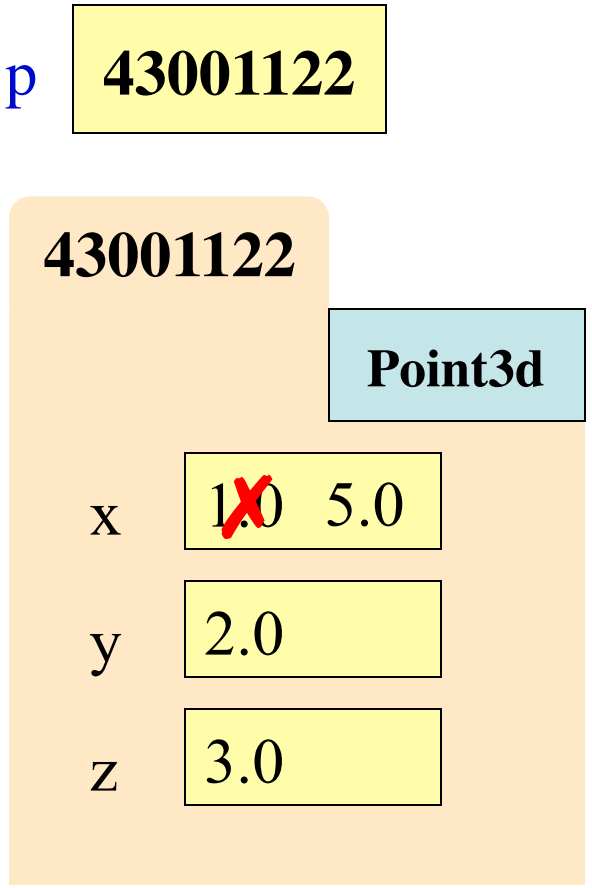
- Variable stores object name
  - **Reference** to the object
  - Reason for folder analogy
- Assignment uses object name
  - **Example**: q = p
  - Takes name from p
  - Puts the name in q
  - Does not make new folder!
- Use `id()` to see folder name
  - `id(p)` evaluates to **43001122**





Objects and Attributes

- Attributes are like variables
  - Can use in expressions
  - Can assign values to them
- **Access**: <variable>.<attr>
  - **Example**: p.x
  - Look like module variables
  - But they are very different
- Putting it all together
  - p.x = p.y + p.z



Exercise: Attribute Assignment

- Recall, q gets name in p

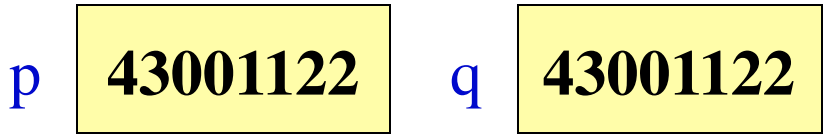
```
p = Point(0,0,0)
q = p
```

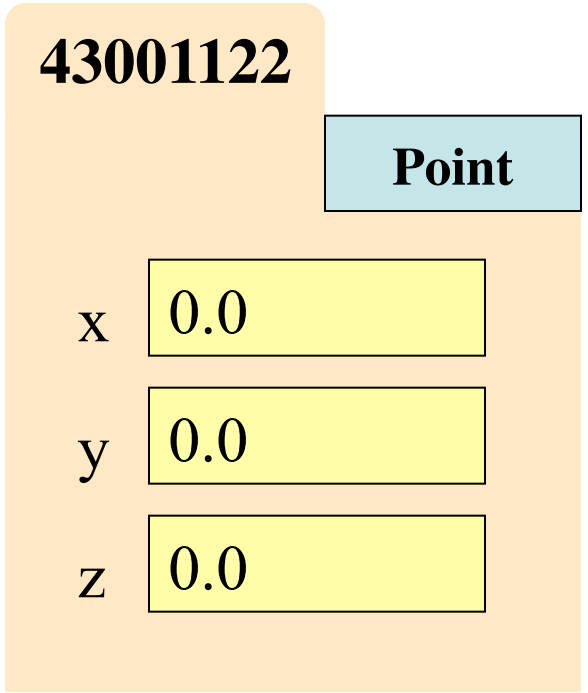
- Execute the assignments:

```
p.x = 5.6
q.x = 7.4
```

- What is value of `p.x`?

A: 5.6
B: 7.4
C: **43001122**
D: I don't know

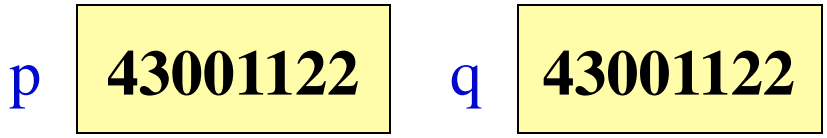




Exercise: Attribute Assignment

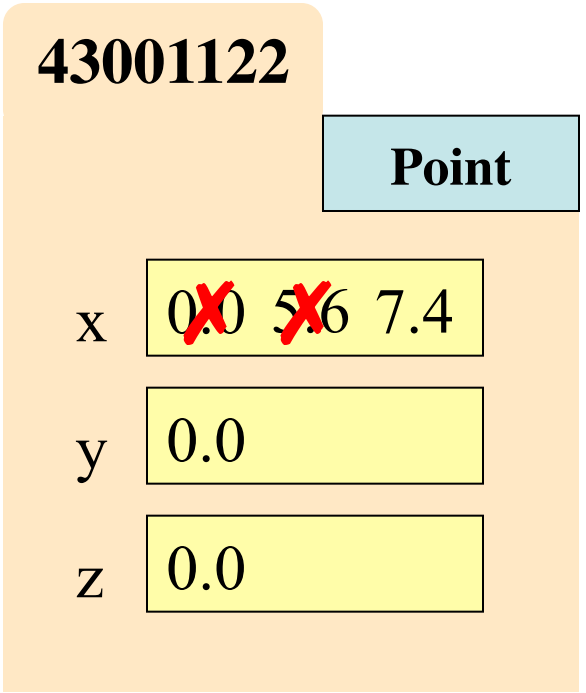
- Recall, q gets name in p

```
p = Point(0,0,0)
q = p
```



- Execute the assignments:

```
p.x = 5.6
q.x = 7.4
```



- What is value of p.x?

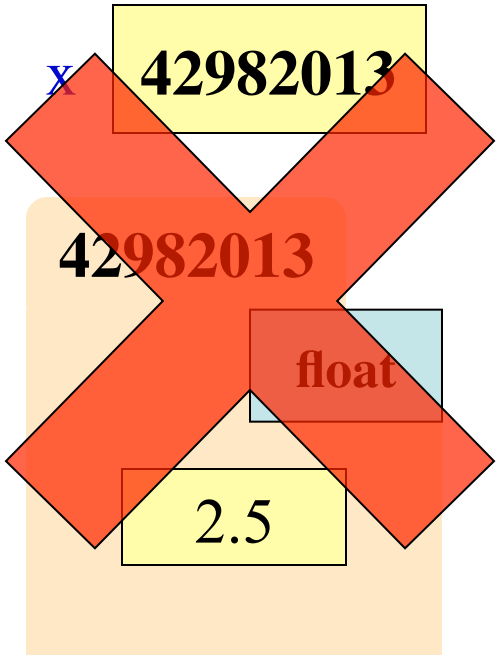
A: 5.6
B: 7.4 **CORRECT**
C: **43001122**
D: I don't know

Surprise: All Values are Objects!

- Including basic values
  - int, float, bool, str
- **Example**:

  ```
  >>> x = 2.5
  >>> id(x)
  ```

- But they are special
  - Have **no named attributes**
  - They are **immutable** (contents cannot change)
  - So we can ignore folder

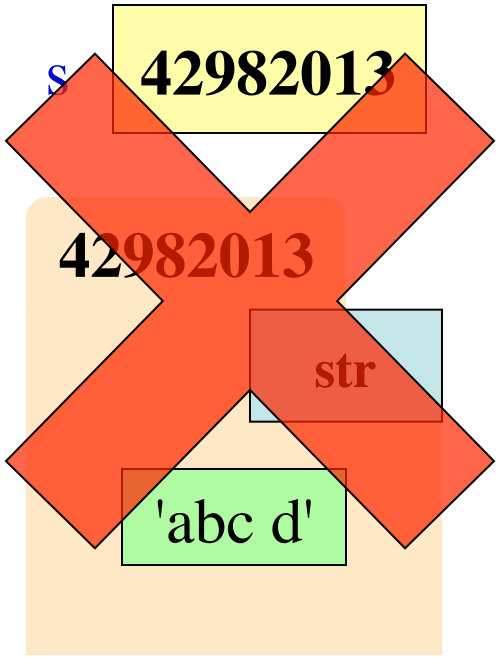


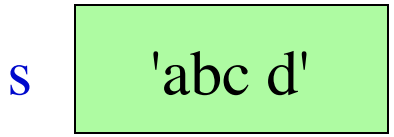
Surprise: All Values are Objects!

- Including basic values
  - int, float, bool, str
- **Example**:

  ```
  >>> x = 2.5
  >>> id(x)
  ```

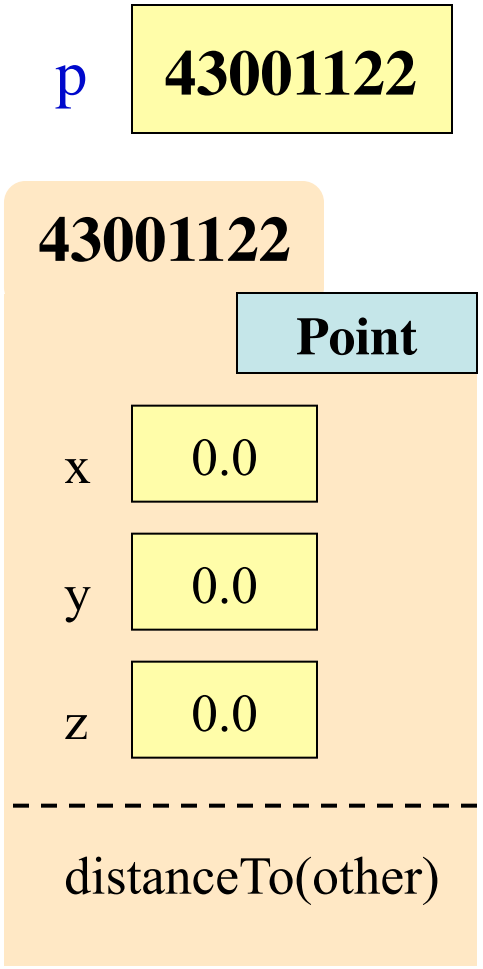
  includes strings
- But they are special
  - Have **no named attributes**
  - They are **immutable** (contents cannot change)
  - So we can ignore folder





Methods: Functions Tied to Objects

- **Method**: function tied to object
  - Has a function call part: <function-name>(<arguments>)
  - But prefix it with variable name: <object-variable>.<function-call>
  - Use of a method is a *method call*
- **Example**: p.distanceTo(q)
  - Both p and q act as arguments
  - Computes distance between two
- Why do it like this? **Later…**



Strings Have Methods Too

```
s = 'Hello World!'
```

See Python API for more

- find(sub)
  - Return the position of substring sub
  - Return -1 if substring not found
  - s.find('o') evaluates to 4
- replace(old, new)
  - Returns a new string; **original is unchanged**
  - Replaces all substrings old with new
  - s.replace('o','uh') evaluates to 'Helluh Wuhld!'

Where To From Here?

- OO Programming is about **creating classes**
  - Eventually you will make your own classes
  - But we need to learn other basics first
- Right now, just try to understand **objects**