

TOPICS

- represent knowledge with data structures
- iteration and recursion as computational metaphors
- abstraction of procedures and data types
- organize and modularize systems using object classes and methods
- different classes of algorithms, searching and sorting
- complexity of algorithms

.0001 LECTURE 1

PYTHON PROGRAMS

- a program is a sequence of definitions and commands
 - definitions evaluated
 - commands executed by Python interpreter in a shell
- commands (statements) instruct interpreter to do something
- can be typed directly in a shell or stored in a file that is read into the shell and evaluated
 - Problem Set 0 will introduce you to these in Anaconda

5.0001 LECTURE 1 23

OBJECTS

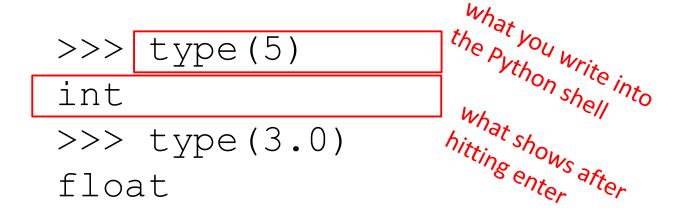
programs manipulate data objects

- objects have a type that defines the kinds of things programs can do to them
 - Ana is a human so she can walk, speak English, etc.
 - Chewbacca is a wookie so he can walk, "mwaaarhrhh", etc.
- objects are
 - scalar (cannot be subdivided)
 - non-scalar (have internal structure that can be accessed)

.0001 LECTURE 1

SCALAR OBJECTS

- int represent integers, ex. 5
- float represent real numbers, ex. 3.27
- bool represent Boolean values True and False
- NoneType special and has one value, None
- can use type() to see the type of an object



6.0001 LECTURE 1

TYPE CONVERSIONS (CAST)

- can convert object of one type to another
- float (3) converts integer 3 to float 3.0
- int(3.9) truncates float 3.9 to integer 3

6.0001 LECTURE 1

OPERATORS ON ints and floats

```
    i+j → the sum
    i-j → the difference if both are ints, result is int if either or both are floats, result is float
    i*j → the product
    i/j → division result is float
```

- i%j → the remainder when i is divided by j
- $i**j \rightarrow i$ to the power of j

6.0001 LECTURE 1 29

SIMPLE OPERATIONS

- parentheses used to tell Python to do these operations first
- operator precedence without parentheses

```
***/
```

+ and – executed left to right, as appear in expression

BINDING VARIABLES AND VALUES

equal sign is an assignment of a value to a variable name

$$variable$$
 $variable$
 $value$
 $variable$
 $value$
 val

- value stored in computer memory
- an assignment binds name to value
- retrieve value associated with name or variable by invoking the name, by typing pi

6.0001 LECTURE 1

ABSTRACTING EXPRESSIONS

- why give names to values of expressions?
- to reuse names instead of values
- easier to change code later

```
pi = 3.14159
radius = 2.2
area = pi*(radius**2)
```

PROGRAMMING vs MATH

in programming, you do not "solve for x"

```
pi = 3.14159
  radius = 2.2
  # area of circle
                                                                                                     an assignment on the right, evaluated to a value

* expression on the right,

* expression of the right,

* expres
                                                                                                                        * variable name on the left

* variable name expression to radius

* equivalent expression

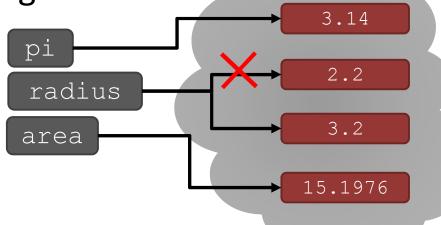
is radius
 area = pi*(radius**2)
  radius = radius+1
                                                                                                                  * Variable name on the left
                                                                                                                                                       is radius += 1
```

CHANGING BINDINGS

- can re-bind variable names using new assignment statements
- previous value may still stored in memory but lost the handle for it

 value for area does not change until you tell the computer to do the calculation again

```
pi = 3.14
radius = 2.2
area = pi*(radius**2)
radius = radius+1
```



6.0001 LECTURE 1 34