

Python Basics

Python Types

- Every Python value has a *type* that describes what sort of value it is
- Built-in function **type** will tell you the type of an expression

English	Python
integer	<code>int</code>
“real” number	<code>float</code>
picture	<code>Picture</code>
pixel	<code>Pixel</code>
colour	<code>Color</code>
string of letters	<code>str</code>

2

Assignment Statement

Form:

variable = expression

How it's executed:

1. Evaluate the expression on the RHS.
(The value of the expression is a memory address.)
2. Store that memory address in the variable on the LHS.

3

Assignment Statement

Form:

Remember this! ***variable = expression***

How it's executed:

1. Evaluate the expression on the RHS.
(The value of the expression is a memory address.)
2. Store that memory address in the variable on the LHS.

Assignment vs Equality

- Python variables look like math variables.
- This could be Python or math:
 $p = 5$
 $q = p * 7$
- But “=” in math means equality
(stating a fact)
whereas “=” in Python means assignment
(asking Python to *do* something)
- This makes a big difference!

5

1. Changeability

- In math, this is inconsistent:
 $p = 5$
 $q = p * 7$
 $p = q + 10$
- p can't be both 5 and 45!
- But in Python, it makes perfect sense. p starts out referring to 5, but then changes to refer to 45.
- You can change a variable's value as many times as you want. You can even change its type.

6

- In math, this makes no sense either:
 $x = x + 1$
It *can't* be true!
- But in Python, it makes perfect sense.
It is asking to make x refer to a something that is one bigger.
- We say “ x is assigned $x + 1$ ” or “ x gets $x + 1$ ”
- Programming languages usually have different symbols for assignment and equality.
Python uses “==” for equality.

7

2. Can't tie two variables

- What does this do?
 $x = 37$
 $y = x + 2$
y is now 39.
 $x = 20$
Is y now 22?
- You can't use assignment to tie the values of two variables together permanently.

8

3. Assignment is not symmetric

	In math	In Python
$\text{sum} = a + b$	they mean the same thing	fine
$a + b = \text{sum}$		illegal

9

Naming

10

Rules for the format of names

- There are a few rules about names of variables:
 - Must start with a letter (or underscore).
 - Can include letters, digits, and underscores, but nothing else.
 - And case matters, by the way.

```
age = 11  
aGe      # Error! This is not defined.
```
- Valid: `_moo_cow`, `cep3`, `I_LIKE_TRASH`
- Invalid: `49ers`, `@home`

11

Conventions for the format of names

- thEre'S a GoOD rEasON wHy WorDs haVE A StaNDaRd caPiTaLiZAtIon sCHemE
- Python convention: `pothole_case`
- **CamelCase** is sometimes seen, but *not for functions and variables*
- Rarely, single-letter names are capitalized: `L`, `X`, `Y`

12

Choosing good names

- Python doesn't care about the *content* of the names, only their format.
- For example, these are equally fine names to Python: `xx3`, `class_average`, `fraggie`
- But we choose names that will be meaningful to the humans who will read our code.
- Eg, if you are adding something up, `sum` or `total` is better than `x`.
- You will be graded on the names you pick.

13

Expressions vs Statements

- English expressions:
 - “The Prime Minister’s wife”
 - “The recycling”
 - “lunch”Each refers to something.
- English sentences:
 - “The Prime Minister’s wife ate pancakes.”
 - “Take the recycling out, please.”
 - “Is it time for lunch?”Each states a fact, asks a question, or gives a command.

14

- Python is similar ...
- Python expressions:
 - `f(x+3)`
 - `98.6 * 2`Each refers to a value.
- Python sentences (“statements”):
 - `temperature = 98.6`
 - `return (x + y + z) / 3`Python statements are always commands to do something (never statements of fact, or questions).

15

Producing textual output

- In Python, you normally make full statements, eg:
 - assignment statements
 - def statements
 - if statements
- But the shell lets you give just an expression, and it then shows you the value of the expression.
- So to show output in the shell, you can just give an expression. 16

- To show output in the editor, use print.

Example:

```
print "Hello!"  
mark1 = raw_input("First mark: ")  
mark2 = raw_input("Second mark: ")  
print "Average:", average(mark1, mark2)
```

- Comma is for printing lists of items, separated by blanks.

17

Getting textual input

- raw_input is a function that:
 - “prompts” the user to type something by printing a string (if you give it one),
 - waits until the user hits the enter/return key,
 - returns a string containing whatever the user typed before hitting enter/return.
- So the value of a raw_input expression is what the user entered.
- Called “raw” because it’s untreated.

18