Python Variables

The 7 Basics of Computer Programming

	Concept	Example from Math
1.	Variables	$x = 5$ *hellothere* = "howdy"
2.	Math & Logic	$5 * 7 + a - 3 / b \% 4$ a is 5 AND $x < 7$ OR $degree \geq 98$
3.	Input/Output (IO)	print "Hello World"

4.	Conditionals	**if** ($x == f(x)$)
		  then print "x is 0 or 1"
		  else print "x is not 0 or 1"

5.	Loops	**foreach** x **in** ($array$) print x
6.	Functions	$f(x) = x^2$
7.	Lists	$array = 1{:}5$ $array = 1, 4, 7, 8, a, b, c, d$

Joy of coding

Data Types

Numbers, Booleans, & Strings

Joy of coding

Numeric Primitive Types

Python Data Type	Description	Examples
`int`	Plain integers (32-bit precision)	-214, -2, 0, 2, 100
`float`	Real numbers	.001, -1.234, 1000.1, 0.00, 2.45
`complex`	Imaginary numbers (have real and imaginary part)	1j * 1J → (-1+0j)

How big (or small/precise) can we get?

- Computer cannot represent all possible values
- Problem: memory has a **finite** capacity
  - The computer only has so much memory that it can devote to each value.
  - Eventually, reach a cut-off
    - Limits size of value
    - Limits precision of value

PI has more decimals, but we're out of space!

0	0	0	0	0	3	.	1	4	1	5	9	2	6	5

Example: in Python interpreter, .1 + .1 + .1 yields 0.30000000000000004.

Strings: `str`

- Indicated by double quotes " " or single quotes ' '
- Treat what is in the " " or ' ' literally
  - Known as **string literals**
- Examples:
  - "Hello, world!”
  - 'c'
  - "That is Buddy's dog."

Single quote must be inside double quotes

Joy of coding

Booleans: `bool`

- 2 values
  - `True`
  - `False`

- More on these when we discuss conditions

What is the value's type?

Value	Type
52	
-0.01	
5.0	
4+6j	
"3.7"	
4047583648	
True	
'false'	

Joy of coding

What is the value's type?

Value	Type
52	int
-0.01	float
5.0	float
4+6j	complex
"3.7"	str
4047583648	int
True	boolean
'false'	str

Joy of coding

Naming Variables

Variable Names/Identifiers

- Variables save data/information
- Variables have names, called ***identifiers***
- An identifier can be any one word that:
  - Consists of letters, numbers, or _
  - Does not start with a number
  - Is not a Python reserved word
    - Examples: `for while def`
- Python is case-sensitive:
  - change isn't the same as Change

Joy of coding

Variable Name Conventions

- Variables start with lowercase letter
- Convention: Constants (values that won't change) are all capitals
- Example: Variable for the current year
  - currentYear
  - current_year
  - CURRENT_YEAR
  - ~~currentyear~~ Harder to read
  - ~~current year~~ No spaces allowed

Naming doesn't matter to computer.
Matters to humans

Joy of coding

Importance of Variable Naming

- Helps you remember what the variable represents
- Easier for others to understand your program
- Examples:

Info Represented	Good Variable Name
A person's first name	`firstName, first_name`
Radius of a circle	`radius`
If someone is employed or not	`isEmployed`

Variable names should be related to the values they store

Variables

Putting it all together

Modeling Information

What data type best represents the info?

Info Represented	Data Type	Variable Name
A person's salary		
Sales tax		
If item is taxable		
Course name		
Gender		
Graduation Year		

Modeling Information

What data type best represents the info?

Info Represented	Data Type	Variable Name
A person's salary	Integer or float	`salary`
Sales tax	Float	`salesTax`
If item is taxable	Boolean	`isTaxable`
Course name	Str	`course_name`
Gender	Str, boolean	`gender, isMale`
Graduation Year	Int	`gradYear`

Just suggestions,
Many other possible variable names

Joy of coding

Assignment Statements

- Variables can be given any value using =
  - Syntax: <variable> = <expression>
  - Semantics: <variable> is set to value of <expression>
- After a variable is set to a value, the variable is said to be *initialized*
- Examples:

```
month = 1
impt_num = 4.5
monthName = 'January'
```

These are **not** equations!
Read "=" as "is set to"

Joy of coding

Assignment Statements

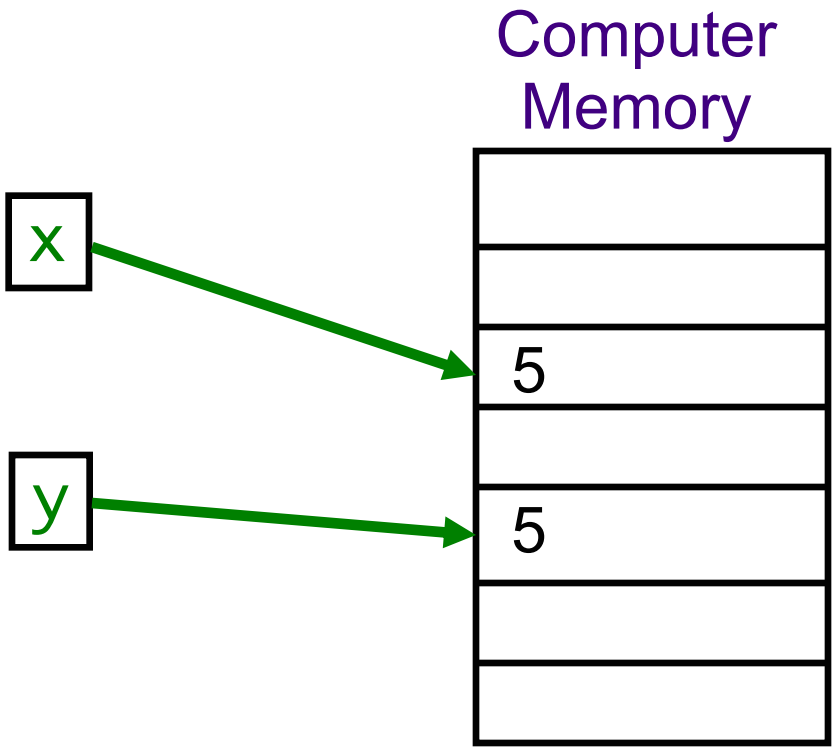
```
x = 5
y = x
```

- Statements execute in order, from top to bottom
- Value of x does not change because of second assignment statement

Assignment Statements

```
x = 5
y = x
```

Does a "lookup" in memory to find value of x



- Statements execute in order, from top to bottom
- Value of x does not change because of second assignment statement

Variables: The Rules

- Only the variable(s) to **left** of the = for the current statement change
  - We'll usually only have one variable on the left

- **Initialize** a variable **before** using it on the right-hand side (rhs) of a statement

Literals

- Pieces of data that are not variables are called ***literals***
- In other words, values you can see in programs
  - Variables point to literal values stored in memory
- Examples:
  - `4`
  - `3.2`
  - `'q'`
  - `"books"`