CS 1340:Fall 2020:Lecture 05

Intro to Python for CS and Data Science

Mark Fontenot, PhD

Southern Methodist University

Variables and Expressions - The Highlights

Assigning a Value to a Var

```
x = 10
y = 'CS 1340 - Intro to CS and DS'
```

- looks like math `=`, but different
- `=` is for **assignment**
- the **left hand side** of `=` must be a variable.
- the **right hand side** of `=` would be an expression
- How we would say it:
  - *set x to 10*
  - *assign x the value of 10*
  - *put the course name in y*

Identifiers

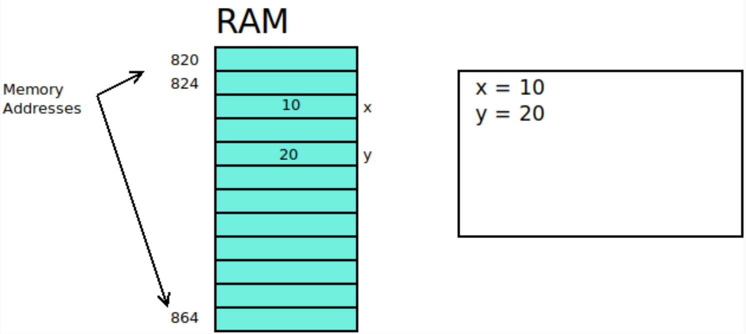
- FYI - for now, we're talking about variable names, but are a little more broad than that
- Rules for identifiers:
  - must start with a **letter** or **underscore**
  - can contain **letters**, **underscores**, and **digits**
  - are **case sensitive**
    - `cat_name` is different from `Cat_NaMe`
  - Certain **reserved words** cannot be used as identifier names
    - examples: `and`, `True`, `None`, `in`, etc. See ZyBooks section 2.2 for complete list
  - cannot contain spaces

Identifiers (Cont'd.)

- Conventions:
  - don't start and end an identifier with double underscore
    - no no! `__init__`
    - technically legal, but Python gives sometimes uses them with special meanings
  - Identifiers should be descriptive
  - all lowercase
  - separate words with underscores
    - `final_grade`, `high_temp`, etc.
- Where do the conventions come from?
  - PEP 8 - Style Guide for Python Code
  - PEP stands for **Python Enhancement Proposal**

Under the Hood

- Where are variables stored while your program is running?



```
x = 10
y = 20
```

Under the Hood - 2

- **object** - represents a value & automatically created by the interpreter when it executes a line of code
- Everything in Python is an **object**
- When you no longer need an object anymore, the **garbage collector** comes behind and *takes out the trash.*

Objects

- **Name Binding** - giving a name to an object in memory
  - an object can have multiple names
  - a name only refers to one object at any point in execution
  - Example: `x = 3`... `x` can never be 3 and 4 at the same time
    - Said another way: `x` can never be referring to two different objects at the same time.

Properties of All Objects

1. **Value** - the current value of the object… IOW, that data associated with it
2. **Type** - the data type of the object (int, string, etc.)
    - **Immutable** - changing the value results in a new object being created (examples: ints and strings)
    - **Mutable** - value of the object can be changed directly
3. **Identity** - a unique identifier for the object
    - Usually refers to the memory address of where the object is stored.
    - We won't worry about this much
    - `id()` function will return the id

Number Time

FP Numbers

- FP = **floating point**
    - number that has a fractional component
- FP Literal
    - include the fractional component even if it is 0.
    - `10.3`, `3.14159`, `10.0`
- For really small or really big things, you can use **scientific notation**
    - Use `e` to precede the exponential part
    - Examples:
        - `avogadros_number = 6.022e23`
        - `plancks_constant = 6.63e-34`

Too Big, Too Small, Too Precise

- Computers cannot store infinitely large or small numbers.
- **Overflow**: when you try to store a number that is too large
  - Example: `2.0**1024`
- Some numbers have too many numbers aver the decimal point
  - Example: `pi = 3.14159`, an approximation of pi.
  - The computer can only represent a finite amount of precision.
- Formatting decimal places when printing:
  - `print('{:.2f}'.format(myFloat))` prints only 2 decimal places of the `myFloat` variable.
  - More details on this as we progress

Expressions

- already talked about this some
- **expression** evaluates to a value
- Evaluation happens following **precedence rules**
    1. ( )
    2. `**` : Exponent
    3. `-` : Unary negation
    4. `*  /  %` : (% is like remainder of division)
    5. `+  -`
    6. left - to - right

Have Some Style, Please!

- use a single blank space before and after an operator
  - NO: `final_average=sum/4`
  - YES: `final_average = sum / 4`
  - Makes your code easier to read
- An Exception: unary negation… don't add a space after it.

Compound Assignment

- uses the current value of a variable, applies an operation to it, stores result back in that variable
- example: `test_grade += 10` will add 10 to whatever is already in `test_grade` and store the result back in `test_grade`
- `+=`
- `-=`
- `*=`
- `/=`
- `%=`

```
test_grade += 10
```

```
test_grade = test_grade + 10
```

No Commas

- don’t put commas in numbers, period.
- NO: `2,000,000`
- YES: `2000000`

Dividing the Pie

- 2 types of division
  - **FP division** - single / - returns floating point result of division
  - **Integer Division** - double '//' - returns only the integer portion of the result.
    - Said another way: returns the floor of FP division
    - `10 // 3` is 3.
    - `10 / 3` is 3.33333333333333
- Reminder, you can't divide by zero and neither can your computer.

The Leftovers

- `%` - the **modulo operator**
  - returns the remainder of division of two integer operands
  - `10 % 5 = 0`
  - `10 % 3 = 1`
  - `3 % 10 = 3`