To Ruby From Python

Python is another very nice general purpose programming language. Going from Python to Ruby, you'll find that there's a little bit more syntax to learn than with Python.

Similarities

As with Python, in Ruby,…

- There's an interactive prompt (called `irb`).
- You can read docs on the command line (with the `ri` command instead of `pydoc`).
- There are no special line terminators (except the usual newline).
- String literals can span multiple lines like Python's triple-quoted strings.
- Brackets are for lists, and braces are for dicts (which, in Ruby, are called "hashes").
- Arrays work the same (adding them makes one long array, but composing them like this `a3 = [ a1, a2 ]` gives you an array of arrays).
- Objects are strongly and dynamically typed.
- Everything is an object, and variables are just references to objects.
- Although the keywords are a bit different, exceptions work about the same.
- You've got embedded doc tools (Ruby's is called rdoc).
- There is good support for functional programming with first-class functions, anonymous functions, and closures.

Differences

Unlike Python, in Ruby,…

- Strings are mutable.
- You can make constants (variables whose value you don't intend to change).
- There are some enforced case-conventions (ex. class names start with a capital letter, variables start with a lowercase letter).
- There's only one kind of list container (an Array), and it's mutable.
- Double-quoted strings allow escape sequences (like `\t`) and a special "expression substitution" syntax (which allows you to insert the results of Ruby expressions directly into other strings without having to `"add " + "strings " + "together"`). Single-quoted strings are like Python's `r"raw strings"`.
- There are no "new style" and "old style" classes. Just one kind. (Python 3+ doesn't have this issue, but it isn't fully backward compatible with Python 2.)
- You never directly access attributes. With Ruby, it's all method calls.
- Parentheses for method calls are usually optional.
- There's `public`, `private`, and `protected` to enforce access, instead of Python's `_voluntary_` underscore `__convention__`.
- "mixins" are used instead of multiple inheritance.

- You can add or modify the methods of built-in classes. Both languages let you open up and modify classes at any point, but Python prevents modification of built-ins — Ruby does not.
- You’ve got `true` and `false` instead of `True` and `False` (and `nil` instead of `None`).
- When tested for truth, only `false` and `nil` evaluate to a false value. Everything else is true (including `0`, `0.0`, `""`, and `[]`).
- It’s `elsif` instead of `elif`.
- It’s `require` instead of `import`. Otherwise though, usage is the same.
- The usual-style comments on the line(s) *above* things (instead of docstrings below them) are used for generating docs.
- There are a number of shortcuts that, although give you more to remember, you quickly learn. They tend to make Ruby fun and very productive.
- There’s no way to unset a variable once set (like Python’s `del` statement). You can reset a variable to `nil`, allowing the old contents to be garbage collected, but the variable will remain in the symbol table as long as it is in scope.
- The `yield` keyword behaves differently. In Python it will return execution to the scope outside the function’s invocation. External code is responsible for resuming the function. In Ruby `yield` will execute another function that has been passed as the final argument, then immediately resume.
- Python supports just one kind of anonymous functions, lambdas, while Ruby contains blocks, Procs, and lambdas.