6
More Python

Terminal I/O

- Python has the usual 3 suspects when it comes to I/O streams: stdin, stdout, stderr.
- Stream objects in Python are called file objects since they are objects of type file.
- Streams can also be redirected like in Perl using the open() command.
- The redirecting is done using the > operator preceded by a file descriptor number: 0 for stdin, 1 for stdout, and 2 for stderr.
- Ex: StdErr.py 2> length_long redirects the stderr in this script to a file named length_long

File I/O

- File I/O is done using the open() function. First parameter is the path to the file, and Second parameter is the mode.
- Modes are r for reading, w for writing, a for appending, and r+ for reading and writing.
- The print>>() or “print chevron” can be used to write an arbitrary number of items to a file. The first argument is the file object and all remaining arguments are written to this file.

File I/O

- The file object supports the following operations (called methods):
  - read(number_of_bytes): reads bytes from a file, or the entire file at once if number_of_bytes is omitted.
  - readline(): reads one line at a time and returns a string for each line. Retains the newline character at the end of a line.
  - readlines(): reads all the lines from a file at once and returns a list of strings.
  - write(string): writes out string to the stream object attached to a file.
  - writelines(sequence_of_strings): writes out a sequence of strings, typically a list.
  - seek(offset_in_bytes): moves the file pointer to the place specified by offset_in_bytes.
  - close(): closes the file. It is not an error to close a file more than once. But it will result in an exception called a ValueError to call a method on a closed file that is only intended to be invoked on an open file.

User-defined Functions

- Docstrings - first line of a function, used to document a function's intended purpose. Should be concise, begin with an uppercase and end with a period.
- Functions are objects just like any other object. They can be assigned to variables, compared to other objects, passed to other functions, etc.
- Functions in Python always return a single object, even if that is the null object called None.
- Unlike in Perl, a function in Python must use the return keyword to return a value from a function. In Perl the last statement is what the function returns.
- When a Python function does not explicitly return any value, it implicitly returns the null object None.

Function Parameters

- Default arguments allows some parameters to be optional in that when omitted, their default values are assigned instead.
- Functions can have an arbitrary number of arguments provided that a parameter is prefixed with a ‘*’. The ordering of parameters matters!
- Functions can have keyword arguments - these arguments use the same syntax as default arguments but can be placed in any order.
- The correct order is ‘regular’ parameters first, then non-keyword arguments, followed by keyword arguments:
- Example of correct ordering: def foo(p, q, *s, **t)

Control Flow

- if - conditional evaluation of a block of code
- while and for - iterative but also conditional
- The else keyword can also be used with while and for.
- for..else will execute a block of code after all the iteration of the for loop in completed.
- while..else will execute a block of code when the conditional of the while loop is false.
- continue - skips the rest of the currently executing loop body and goes to the next iteration.
- break - causes control flow to exit the current loop.

Truthiness and Falsiness in Python

- Every object in Python can be evaluated for its truth value.
- Any two objects can be compared using the relational operators <, <=, >, >=, and the equality operators == and !=.
- Class instances (objects) are considering ordered by their memory addresses.
- Objects of different types are ordered by their typenames when compared using relational comparisons.
- Operators that test for identity - ‘is’ and ‘isnot’
- Operators that test for membership - ‘in’ and ‘not in’

Python Modules

- Like Perl, Python supports the concept of modules.
- Modules contain closely related code (classes, functions, etc)
- Module names are the name of the .py file without the .py extension.
- All Python code executes within the context of a module, even when no module name is explicitly mentioned.
- The scripts we have been running are executed in the “top-level” implicit module __main__.
- Modules can be imported into a script using the import statement.
- It is also possible to import names from a module without importing the entire module.

Python Scope

- A code block in Python can be defined by any of these objects:
  - Script file
  - Module
  - Function Body
  - Class definition
- Names of objects are “bound” to a local namespace (code block) when any name binding operation takes place, and have local scope within that block.
- Name binding operations include things like defining a function or class or naming a parameter in a function definition.

Python Scope

- Every code block in Python has its own namespace.
- A global variable in Python is bound to the module level scope. This often is the same level of scope as the script file or the __main__ namespace.
- Python is statically or lexically scoped. This means that all decisions as to what namespace a given name should be resolved in are made at compile time.
- If a name cannot be resolved in the namespace of the current scope, the compiler will try to resolve it in the namespace of the nearest enclosing scope that has a binding for that name.

Reading and Watching Assignments

- https://developers.google.com/edu/python/ - Read the following sections:
  - Overview
  - Python Setup
  - Introduction
  - Strings
  - Lists
  - Sorting
  - Dicts and Files
- Watch the first 3 videos:
  - Introduction, strings - https://www.youtube.com/watch?v=tKTZoB2Vjuk
  - Lists and sorting - https://www.youtube.com/watch?v=EPYupizJYQI
  - Dicts and files - https://www.youtube.com/watch?v=haycL41dAhg