Scripting Languages

CIS*2750

Advanced Programming Concepts

Scripting Languages

- Scripting languages are generally **interpreted** instead of **compiled**.
  - Immediate execution allows for **rapid development** and change.
- "Easy to learn" and have support for high-level structures and libraries.
  - Much of the initial work for many applications has been done and is available in libraries.
  - Complex data structures such as lists and dictionaries usually built in

Scripting Languages

- **Python** uses combination of compilation and interpretation:
  - The source code is converted to an intermediate form called **byte code** in a step similar to compilation.
  - The byte code is executed by an interpreter.
  - This improves performance over purely interpreted systems.

Source → *Compiler Linker* → Executable → *"Bare metal" processor* → Execution

Compiled Execution

Source → *Interpreter* → Execution

Interpreter Execution

Source → *Compiler* → Byte Code → *Interpreter* → Execution

Byte Code Execution

Scripting Languages

- Scripting languages are often used to **combine** the functionality of other programs.
  - They act as *glue*.
- This allows the script to act as the intermediary between programs and pass information between them.

Extending

- External programs can be used to increase the functionality of the scripting language by binding existing programs to it
  - This is called **extending**.
- This allows function calls to be made directly to the compiled programs instead of through command interface (**stdin/stdout**)

Dynamic Typing

- Scripting languages often support **dynamic typing**.
- The system manages the types of variables without the programmer's explicit input on matters of length and type declarations.
- Some languages treat all variables as *strings* and reinterpret them (say, as numbers) when non-string operations occur with the variable.

Dynamic Typing

- This can lead to type mismatches when a variable is assigned a type that is not expected!
- It can also cause problems if a variable name is simply **misspelled**!
  - Results in *two* different variables existing when only *one* was intended.

Memory Management

- Automatic memory management controls the allocation and freeing (*garbage collection*) of memory on demand.
  - Objects can grow and shrink as needed and are removed when no longer necessary.

Object-Oriented

- Many scripting languages have adopted object-oriented structures.
- Traditional scripting languages tend to become difficult to manage when used to write larger programs and the inclusion of OO is an attempt to address the problem.
  - E.g. Python, incrTCL, Object Oriented Perl

OO and Python

- Python supports most OO concepts:
  - Ability to create multiple name spaces (scope).
    - Each object contains its own name space.
  - Polymorphism
    - Methods called vary based on argument's class.
  - Operator Overloading
    - Operators given multiple meanings.
  - Multiple Inheritance
    - A class can be the product of multiple parents.

Dynamic Code Creation

- Many scripting languages can dynamically create and execute code during the execution of the script.
  - This generally cannot be done by a non-scripting language.

```
a = 10
x = “print a”
exec(x)
```

Data Structures

- Most modern scripting languages have built-in support for high level data structures.
  - Associative arrays
    - Also called **dictionaries** or **hash** tables.
  - Lists