CSCI 334:
Principles of Programming Languages

Lecture 20: Scope

Instructor: Dan Barowy
Williams

Outline

What is scope?

What are the kinds?

Why is it important?

How does it work?

Your to-dos

1. Lab #10, **due Sunday 12/3**
2. Want to talk about your project?
   Office hours today 4-5pm.

Final project timeline

1. ~~Project proposal (Lab 8), **due Sun 11/12**~~
2. ~~Minimally working version (Lab 9), **due Sun 11/19**~~
3. Language specification doc (Lab 10), **due Sun 12/3**
4. Mostly working version (Lab 11), **due Sun 12/10**
5. Project + video presentation (Lab 12), **due Sun 12/17**

Scope

Recall that a **variable** is a named placeholder for a value in an expression. **Scope** is a set of rules that determines **what value** is returned when a variable is used in an expression.

If your language does not have **functions** (or **blocks**, like **for loops**), scope rules are mostly irrelevant.

Kinds

There are two common kinds of scope.

- **Lexical** scope
- **Dynamic** scope

Both definitions depend on a notion of **time**.

- Lexical scope depends on **compile time**.
- Dynamic scope depends on **run time**.

There are many more uncommon kinds of scope, but if you work to understand the two above, you will find it easier to deal with the weirdos.

Importance

Scope rules are used to determine:

- Which **values** are returned.
- When **garbage collection** is run.

Scope rules can have an impact on whether programmers write **buggy** programs. Here are some languages with **surprising** scope rules:

- JavaScript
- R
- LISP (the original)
- Bash
- Mathematica

Dynamic scope

Dynamic scope is a rule that finds the **most recent value** of a given variable in a program's execution (i.e, at **run time**).

Lexical scope

Lexical scope is a rule that uses the **lexically closest value** of a variable at the time the use was defined (i.e., at **compile time**).

Master scope, master programming

Want to be a front-end developer? You should probably know these rules:

https://stackoverflow.com/a/500459/480764

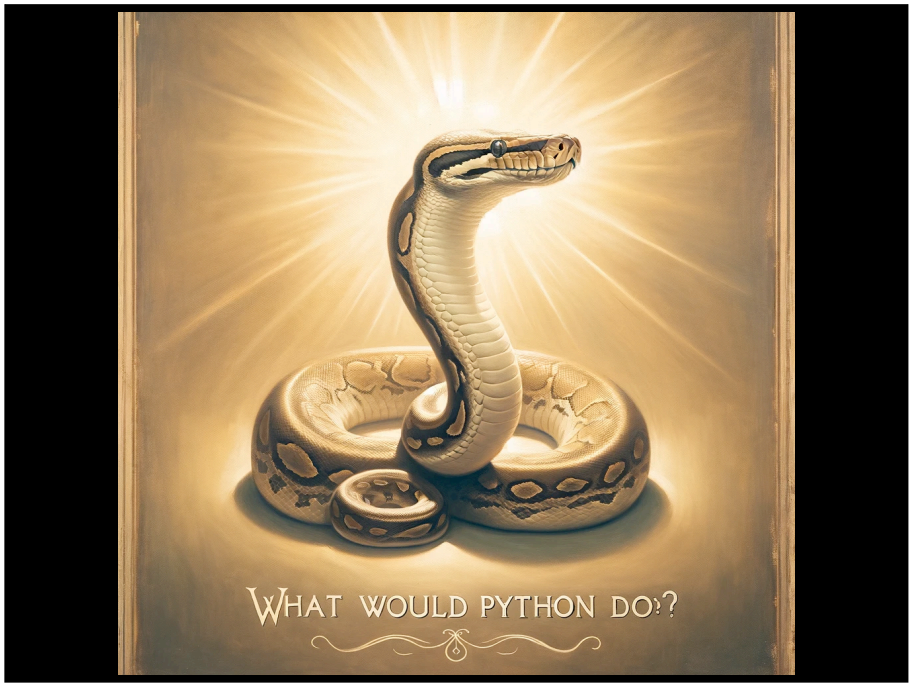
Perl Examples

```
local $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   local $x = 20;

   f();
}

g();
```

```
my $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   my $x = 20;

   f();
}

g();
```

What is printed?

Which one is dynamic and which one is lexical?



Perl Examples

```
local $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   local $x = 20;

   f();
}

g();
```

Dynamic scope
(`local` keyword)

```
my $x = 10;

sub f
{
   print $x."\n";
}

sub g
{
   my $x = 20;

   f();
}

g();
```

Lexical scope
(`my` keyword)

How do they work?

(whiteboard)

How does a function work?

What is the output of this program?

```
def plus(x,y):
  x = x + 1
  return x + y

x = 1
y = 2

print(plus(y,x))
print(x)
print(y)
```

Why?

Recap & Next Class

This lecture:

Scope

Next lecture:

Object-Oriented Programming