CSCI 334:
Principles of Programming Languages

Lecture 18: Scope

Instructor: Dan Barowy

Williams

Outline

Project activity

What is scope?

What are the kinds?

Why is it important?

How does it work?

Your to-dos

1. Lab 10, **due Wednesday 4/30**
2. Want to talk about your project?
   Office hours today 4-6pm.

Final project timeline

1. ~~Brainstorm (Lab 7), **due Wed 4/9**~~
2. ~~Project Proposal (Lab 9), **due Wed 4/23**~~
3. Minimally working version (Lab 10), **due Wed 4/30**
4. Final project + video presentation (Lab 11), **due Wed 5/14**

Important end-of-semester things

1. **Last quiz**: this **Thursday**!
2. Midterm #2 **review**, **Monday, May 5**
3. Midterm **exam** #2, **Thursday, May 8**

Project Activity

Project Activity

Find a partner **who is not your project partner**.

I will prompt you when to move to the next step in the procedure below.

Each of you will do the following in turn:

1. (~4 minutes) **Explain your project** to your partner. Be sure to discuss at least one primitive and one combining form, and be sure to describe the form of the input and the form of the output.
2. (~4 minutes) Your partner **explains your project back to you**. Take note which concepts they have trouble explaining back to you.
3. **Swap roles** and **go to step 1**.

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** can be performed.

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

```perl
local $x = 10;

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

sub g
{
    local $x = 20;

    f();
}

g();
```
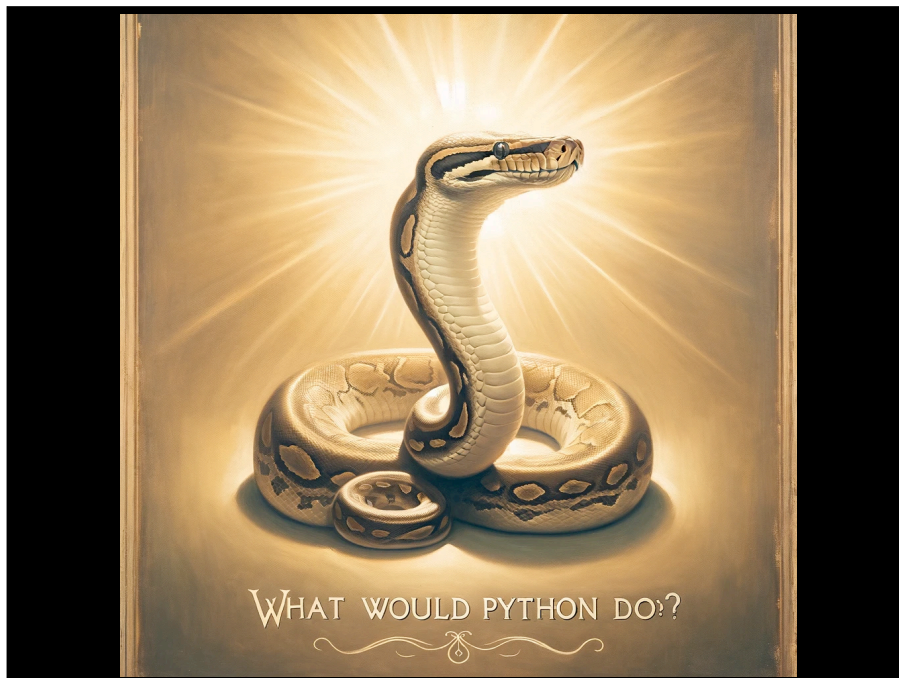
```perl
my $x = 10;

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

sub g
{
    my $x = 20;

    f();
}

g();
```

prints: 20   **prints: 10**

Which one is dynamic and which one is lexical?

WHAT WOULD PYTHON DO?

Perl Examples

```perl
local $x = 10;

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

sub g
{
    local $x = 20;

    f();
}

g();
```

```perl
my $x = 10;

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

sub g
{
    my $x = 20;

    f();
}

g();
```

Dynamic scope   **Lexical scope**
(`local` keyword)   (`my` keyword)

How do they work?

(whiteboard)

How does a function work?

What is the output of this program?

```python
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:

Testing code/How to give a good talk