

Recitation 1: Sly.py

CENG 444 - Language Processors
Fall 2022

PyExp Language

- Task: Convert all of the assignments into TAC-like python code. Convert all of the if-else expressions into if statements.
- Non-python ID = exp if exp is equivalent to ID = exp if exp else None. (added to demonstrate dangling else problem)

```
a = 3
```

```
a = 5+3*(2-a)
```

```
b = a if a == 3 else 0
```

```
c = 5 if a == 3
```

```
a = 3
```

```
t2 = 2 - a
```

```
t1 = 3 * t2
```

```
t0 = 5 + t1
```

```
a = t0
```

```
t4 = a == 3
```

```
if t4:
```

```
    t3 = a
```

```
else:
```

```
    t3 = 0
```

```
b = t3
```

```
t6 = a == 3
```

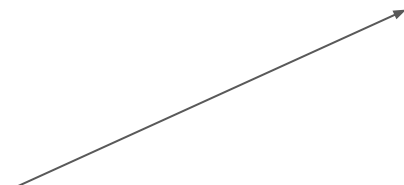
```
if t6:
```

```
    t5 = 5
```

```
else:
```

```
    t5 = None
```

```
c = t5
```



Let's figure out the lexical rules and tokens!

- Identifiers
- Integers
- Binary operators `+`, `-`, `=`, `==`, `*`, `//`
- `if`, `else`
- Anything else? :)

Sly.Lexer

Check `pyexp_lexer.py`, `pyexp_lexer_test.py`,
`pyexp_test_tokens.txt`

Let's figure out the grammar rules!

- Assignment
 - `a = (b = 3)` is a syntax error in Python but `a = b = 3` isn't, weird! However we will allow it for so that the grammar challenges us.
- Expressions: summations, subtractions, multiplications, divisions
- If-else expressions
 - Normally `a = 3 if 3 if 5 else 2 else 3` is not allowed but that makes the grammar too easy for this recitation so we will allow it.
 - How should the parse tree of `a = 3 if 5 if 3` be?
 - What about `a = 2 + 3 if 0 else 2` ?
 - Harder example: `a = 10 if 8 if 9 else 7 if 11`
- Anything else? :)

Sly.Parser

Check `pyexp_parser.py`, `pyexp_parser_test.py`,
`example.pyexp`, `example.pyexp.tree`

Generating Python Code

- We do a DFS on the AST.
- Each DFS call will return two things:
 - **name**: representing the name of the variable where the value obtained from calculation(evaluation) of the node is stored. *We might have to generate it ourselves!*
 - Consider expression $(3+5*4-2)$. We have to store $5*4$ at a temporary variable.
 - **code**: representing *the python code that has to be evaluated before* to calculate the value stored in **name**

Translating to Python Code

Check `pyexp_translate.py`, `example.pyexp`, `example.pyexp.py`

Other Details


- You will learn in class that we don't even have to generate AST for this task and do it in a way called *syntax-directed-translation(SDT)*, because code and name are *synthesized attributes*.
- Might not work for every PL feature! (return/break statements, type checking rules, checking duplicate declarations etc.)
- We also do not have to store code in the nodes (analyze the list held in code variables carefully).

Bonus round: Breakwhile Language!

- Language with mock statements within if and while statements. And a sinister break statement
- Task: Generate labels for break to match their whiles

```
while {  
    If { if { break; }}  
    while { break; }}
```

```
while(label0) {  
    if { if { break out of  
label0; }}  
    while(label1) { break out  
of label1; }}
```



Synchronization Rules for Error Recovery

Check `breakwhile_parser.py`

Matching breaks and whiles

Check `breakwhile_translate.py`

Questions?