Syntax Revisited

Williams College
CS334 - S09
Prof. McGuire

Python Features

- Eager
- Le
  - res
  - ations!
- N
  -
  -
- Array/Table-centric library and syntax

```
(let ([x null])
   (set! x 3)

   (let ([x null])
      (set! x 4)
      x)


   x)
```

Python Syntax

Anonymous closure creation:

lambda [*id* (**,** *id*)*] **:** *exp*

Procedure declaration:

def *id* **(**[id (**,** *id*)*]**):**
exp^{+}

Application:

exp **(** *exp** **)**

Iteration:

for *exp* **in** *exp* **:**
exp^{+}

Conditional:

if *exp* **:**
exp^{+}
[**else:**
exp^{+}]

Assignment:

id **=** *exp*

[*exp* **.**]* *id* **=** *exp*

Return (*only in procedure*):

return *exp*

Spaces and newlines indicate block and expression extent.
Special syntax for constructing arrays, strings, tuples, and tables.

Case Study: a[i]

Java

$$\frac{exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle \qquad exp_i \Rightarrow i \in \{0...n-1\}}{exp_a[exp_i] \Rightarrow v_i}$$

C++

$$exp_a[exp_i] \Rightarrow exp_a\texttt{.operator[](}exp_i\texttt{)}$$

array[5] = true table[“course”] = 334

$$\frac{exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle \qquad exp_i \Rightarrow i \in \{0...n-1\}}{exp_a[exp_i] \Rightarrow v_i}$$

$$\frac{exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle \qquad exp_i \Rightarrow i \notin \{0...n-1\}}{exp_a[exp_i] \Rightarrow \text{unspecified}}$$

(The Array behavior here is for std::vector, which is in the standard library but not part of the C++ language specification. Many programmers use their own array classes, like G3D::Array, that ascribe semantics closer to Java’s)

Python

$$\frac{\neg(exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle)}{exp_a[exp_i] \Rightarrow exp_a\texttt{.get(}exp_i\texttt{)}}$$

$$\frac{exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle \qquad exp_i \Rightarrow i \in \{-n...n-1\}}{exp_a[exp_i] \Rightarrow v_{i \mod n}}$$

if a[-1]=“~”: print a + “is a temp file”

$$\frac{exp_a \Rightarrow Array\langle v_0, v_1, ..., v_{n-1}\rangle \qquad exp_L \Rightarrow i \in \{-n...n-1\} \qquad exp_H \Rightarrow i \in \{-n...n\}}{exp_a[exp_L\,{:}\,exp_H] \Rightarrow Array\langle v_{L \mod n} ... v_{(H-1) \mod n}\rangle}$$

if a[-3:0]=“.jpg”: print a + “is an image file”

(Array is called List in Python)

C

$$\frac{exp_a \Rightarrow a \in \tau \qquad exp_i \Rightarrow i}{exp_a[exp_i] \Rightarrow}$$

Read sizeof(τ) bytes from memory location *a*+sizeof(τ)**i* and interpret them as a value of type τ.