CSC 223 - Advanced Scientific Programming

Basic Python Syntax

Syntax and Semantics

- The *syntax* of a programming language refers to structure of the language, that is, what constitutes a legal program.
- The *semantics* of a programming language refers to the meaning of a legal program.

Example Program

```
# set the midpoint
midpoint = 5

# make two empty lists
lower = []; upper = []

# split the numbers into lower and upper
for i in range(10):
    if (i < midpoint):
        lower.append(i)
    else:
        upper.append(i)

print("lower:", lower)
print("upper:", upper)
```

Comments

- Comments are denoted by #.
- The example program starts with the comment:

```
# set the midpoint
```

- Anything after the # is ignored by the interpreter.
- Python has no syntax for multi-line comments.

The End-of-Line Terminates a Statement

- The next line in the script is:

```
midpoint = 5
```

- This is an assignment operation which binds the name `midpoint` to the value 5.
- The statement is marked by the end of the line
- A statement can span multiple lines by using the backslash:

```
x = 1 + 2 + 3 + 4 +\
    5 + 6 + 7 + 8
```

- or by placing the expression in parentheses:

```
x = (1 + 2 + 3 + 4 +
     5 + 6 + 7 + 8)
```

Semicolon Optionally Terminates a Statement

- The next part of the script is

```
lower = []; upper = []
```

- The semicolon can be used to put multiple statements on the same line.
- This is equivalent to

```
lower = []
upper = []
```

- The use of semicolons in this fashion is generally discouraged in Python programming.

Indentation: Whitespace Matters

- The main block of code in the program is:

```
for i in range(10):
    if (i < midpoint):
        lower.append(i)
    else:
        upper.append(i)
```

- This is a compound control-flow statement.
- Python blocks of code are denoted by indentation.
- Python blocks of code are always preceded by a colon (`:`) on the previous line.
- Most style guides recommend indenting blocks by four spaces.

Whitespace Within Lines Does Not Matter

- The following expressions are equivalent:

```
x=1+2
x  =  1  +  2
x          =          1        +          2
```

- Most style guides recommend using a single space around binary operators and no space around unary operators.

Parentheses

- Parentheses can be used for grouping or calling
- Grouping statements or mathematical operations:

```
2 * (3 + 4)
```

- Calling a function:

```
print('first value:", 1)
```

- Some functions can be called with no arguments; but the parentheses are still needed.

```
L = [4, 3, 2, 1]
L.sort()
print(L)
```