# CSC 223 - Advanced Scientific Programming

Python List Comprehensions and Generators

# List Comprehensions

- A list comprehension is a way to compress a list building for loop into a shorter line of code.
- Example list building for loop

```
L = []
for n in range(12):
    L.append(n ** 2)
```

- The equivalent list comprehension

```
L = [n ** 2 for n in range(12)]
```

- Basic syntax:
  [*expression* for *variable* in *iterable* ]

# Multiple Iteration

- A list can be built from multiple values

```
[(i,j) for i in range(2) for j in range(3)]
```

- This is equivalent to nested for loops; the interior index varies the fastest.

# Conditionals on the Iterator

- A conditional can be added to the end of the expression

```
>>> [val for val in range(20) if val % 3 > 0]
[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19]
```

- This is equivalent to the following loop:

```
L = []
for val in range(20):
    if val % 3 > 0:
        L.append(val)
```

# Conditionals on the Value

- Python has a conditional expression (note, not statement)

```
>>> val = -10
>>> val if val >= 0 else -val
10
```

- This is often used within list comprehensions and `lambda` functions

```
>>> [v if v % 2 else -v for v in range(10)]
[1, -2, 3, -4, 5, -6, 7, -8, 9]
```

# Other Comprehensions

- `set` comprehensions

```
>>> {n ** 2 for n in range(10)}
{0, 1, 4, 9, 16, 25, 36, 49, 64, 81}
```

- `dict` comprehensions

```
>>> {n:n ** 2 for n in range(6)}
{0: 0, 1: 1, 2: 4, 3: 9, 4: 16, 5: 25}
```

- generator expression

```
>>> (n**2 for n in range(12))
<generator object <genexpr> at 0x1027a5a50>
```

# Generators

- A list is a collection of values
- A generator produces values as they are needed
- A generator exposes the iterator interface

```
for val in (n ** 2 for n in range(10)):
    print(val, end=' ')
```

- A generator can only be iterated through once

# Generator Functions

- A generator function makes use of the `yield` statement
- The generator expression

```
G1 = (n ** 2 for n in range(10))
```

  is equivalent to

```
def gen():
    for n in range(10):
        yield n ** 2

G2 = gen()
```