Computational Programming with Python
Unit 8: Iterators

Center for Mathematical Sciences, Lund University

Lecturer: Claus Führer, Alexandros Sopasakis
Iterators
Definition: Iterable objects

Definition
An iterable object has a special method `__iter__` which is called in for loops:

Example
A typical iterator is `range` in Python 3.x:

```python
for i in range(100000000):
    if i > 10:
        break
```

Is the same as

```python
for i in range(100000000).__iter__():
    if i > 10:
        break
```

`range(10000).__iter__()` creates an *iterator*. 
Iterators

Definition
An iterator has a special method \_\_next\_.

Example

```python
rg = range(3)  # iterable object
rgi = rg.__iter__()  # an iterator
rgi.__next__()  # returns 0
rgi.__next__()  # returns 1
rgi.__next__()  # returns 2
rgi.__next__()  # returns StopIteration exception
```

Iterators can be exhausted. They exist as objects but are of no use any more in your program.
Own iterators

Creation of iterators is possible with the keyword `yield`:

```python
def odd_numbers(n):
    "generator for odd numbers less than n"
    for k in range(n):
        if k % 2 == 1:
            yield k
```

Then you call it as:

```python
g = odd_numbers(10)
for k in g:
    ... # do something with k
```
Iterator Tools

- **enumerate** is used to *enumerate* an iterable:

```python
A = ['a', 'b', 'c']
for i, x in enumerate(A):
    print(i, x)  # result: 0 a 1 b 2 c
```

- **reversed** creates an iterator from a list by going backwards:

```python
A = [0, 1, 2]
for elt in reversed(A):
    print(elt)  # result: 2 1 0
```
Infinite Iterators

Iterators can be infinite:

Example

```python
def odd_numbers():
    "generator for odd numbers"
    k=0
    while True:
        k+=1
        if k % 2 == 1:
            yield k

    # Find the first odd_number greater than 68:
on=odd_numbers()

    for nu in on:
        if nu > 68:
            print(nu)
            break
```
Example: Arithmetic Geometric Mean

With \( a_0 = 1 \) and \( b_0 = \sqrt{1 - k^2} \), the iteration

\[
a_{i+1} = \frac{a_i + b_i}{2}
\]

\[
b_{i+1} = \sqrt{a_i b_i}
\]

converges to

\[
F(k, \pi/2) := \frac{\pi}{2} \lim_{i \to \infty} \frac{1}{a_i} = \int_0^{\pi/2} \frac{1}{\sqrt{1 - k^2 \sin^2(\theta)}} \, d\theta
\]

This is called a complete elliptic integral of the first kind.
Example: Arithmetic Geometric Mean (Cont.)

```python
def arithmetic_geometric_mean(a, b):
    """
    Generator for the arithmetic and geometric mean
    a, b initial values
    """
    while True:  # infinite loop
        a, b = (a+b)/2, sqrt(a*b)
        yield a, b
```

```python
def elliptic_integral(k, tolerance = 1e-5):
    """
    Compute an elliptic integral of the first kind.
    """
    a_0, b_0 = 1., sqrt(1-k**2)
    for a, b in arithmetic_geometric_mean(a_0, b_0):
        if abs(a-b) < tolerance:
            return pi/(2*a)
```

C. Führer, A. Sopasakis, Lund University, Sweden
Just as we had list comprehensions, there is also *generator comprehension*:

```python
extendedchars
extendedchars
extendedchars
extendedchars
g = (n for n in range(1000) if not n % 100)
# a generator that generates 0, 100, 200,
```

C. Führer, A. Sopasakis, Lund University, Sweden
A useful Python module in this context is "itertools"