Informator
A SOFRANO COMPANY

GENERATORS

PYTHON

PYTHON

ADVANCED ITERATORS

Iterators are VERY common in the Python language and acts like a foundation for more advanced techniques and principles.

Iteration (iterables and iterators) is the base for

- Sorting
- Reversing
- Transformation of data
- And much much more

PYTHON

ADVANCED ITERATORS

There is two more ways of creating Python iterables that we haven't mentioned yet.

- Create a function that Python can iterate over on its own (__getitem__)
- Create an iterator following the iteration interface (__iter__, __next__/next)
- **Use a generator expression**
- **Create a generator**

LET'S COMPARE

Informator
A SOFRAND COMPANY

Let's have a look at this List Comprehension

```
a = [n+3 for n in range(5)]
```

Compare it with the following:

```
a = list(n+3 for n in range(5))
```

PYTHON

GENEXP

That is a **generator expression** passed to a list constructor.

```
a = list(n+3 for n in range(5))
```

This is the same **generator expression** used for creating a ”generator object”.

```
a = (n+3 for n in range(5))
```

A **generator comprehension** is a generator expression in-between parenthesis.

PYTHON

GENEXP

Generator expressions acts like list comprehensions, but instead of returning a list it returns a generator.

The generator **yields** one value at a time and could save a lot of memory!

The generator follows the iteration protocol that you might heard of :-)

Generator expressions provide an additional shortcut to build generators out of expressions!

PYTHON

A LOT OF CHOICES

```
a = [n+3 for n in range(5)]                   # list comprehension

a = (n+3 for n in range(5))                   # generator expression

def myGeneratorFunction(n):                   # generator function
    ????
```

PYTHON

GENERATORS

Generators simplifies the creation of iterators

A generator is a function that produces a sequence of results instead of a single return-value.

A generator IS an iterator, BUT you don't have to care about the iterator protocol

GENERATORS

Informator
A SOFRAND COMPANY

The word ”generator” is used to describe a function that generates something.

But also what that function actually generates.

We usually talk about ”generator functions” and ”generator objects”.

If someone talk about ”generators” is usually refer to ”generator objects”.

PHew !!

PYTHON

BASIC PRINCIPLES

- When a generator function is called it returns a generator object.
- The generator object is an iterable
- When next() is called it executes the code and yields a value
- The yielded value is returned from the next() function.

GENERATOR FUNCTIONS

Local variables and execution state are automatically saved between calls

```
def myRange(n):
     i = 0
     while i < n:
          yield i
          i += 1


a = myRange(3)
a.next()
a.next()
a.next()
a.next()
```

PYTHON

EXERCISE

1. Create a function that return a list of numbers from 0 to N. Loop through the result and print it to the screen.
2. Create an iterable class. Instances should return numbers from 0 to N when traversed in a loop.
3. Rewrite the first function as a generator function with the yield keyword. Loop through the result and print it to the screen.

All 3 parts of the exercise above return different types of iterables.

They do give the same result when used with a for-loop.

PYTHON

EXERCISE

Create a generator function that summarizes all odd numbers in between 1 and 10.000.000.

GENERATOR ITERATORS

```
class myIterable(object):

   def __init__(self,a,b):
      self.a = a
      self.b = b

   def __iter__(self):
      i = self.a
      while i < self.b:
         yield i
         i+=1
```

PYTHON

GENERATOR ITERATORS

Informator

A SOFRAND COMPANY

Generator function

```
def __iter__(self):
    for i in self.name:
        yield i
```

Generator expression

```
def __iter__(self):
    return (i for i in self.name)
```

PYTHON

EXERCISE

Create an iterable that take one text string as argument.

Return one character at a time from the text string for each iteration.

Use a separate iterator object so you could handle several iterations at the same time.

Use a generator to create your iterator.

Extra:

Iterate over the string by alternate from which direction you grab characters.
For a string with 7 characters you'll grab in the following order: 1,3,5,7,6,4,2