# Exercise: Iterators and Generators

Problems for exercise and homework for the [Python OOP Course @SoftUni](). Submit your solutions in the SoftUni judge system at [https://judge.softuni.bg/Contests/1945](https://judge.softuni.bg/Contests/1945)

## 1. Take Skip

Create a **class** called **`take_skip`**. Upon initialization it should receive a **step** (number) and a **count** (number). Implement the **`__iter__`** and **`__next__`** functions. The iterator should return the **count amount** of numbers (**starting from 0**) and with the **given step**. For more clarification, see the examples:

***Note: Submit only the class in the judge system***

### Examples

| Test Code | Output |
|---|---|
| `numbers = take_skip(2, 6)`<br>`for number in numbers:`<br>`    print(number)` | `0`<br>`2`<br>`4`<br>`6`<br>`8`<br>`10` |
| `numbers = take_skip(10, 5)`<br>`for number in numbers:`<br>`    print(number)` | `0`<br>`10`<br>`20`<br>`30`<br>`40` |

## 2. Dictionary Iterator

Create a class called **`dictionary_iter`**. Upon initialization it should receive a **dictionary** object. Implement the iterator, so it returns **each key-value pair** of the dictionary as a **tuple of two elements** (the key and the value).

***Note: Submit only the class in the judge system***

### Examples

| Test Code | Output |
|---|---|
| `result = dictionary_iter({1: "1", 2: "2"})`<br>`for x in result:`<br>`    print(x)` | `(1, '1')`<br>`(2, '2')` |

## 3. Countdown Iterator

Create a class called **`countdown_iterator`**. Upon initialization it should receive a **count**. Implement the **iterator**, so it returns **each number of the countdown** (from count to **0** inclusive).

***Note: Submit only the class in the judge system***

## Examples

| Test Code | Output |
|---|---|
| `iterator = countdown_iterator(10)`<br>`for item in iterator:`<br>`    print(item, end=" ")` | `10 9 8 7 6 5 4 3 2 1 0` |

## 4. Take Halves

You are given a skeleton with the following code:

```
def solution():
    def integers():
        # TODO: Implement

    def halves():
        for i in integers():
            # TODO: Implement

    def take(n, seq):
        # TODO: Implement

    return (take, halves, integers)
```

Implement the **three** generator functions:

- **`integers()`** - generates an **infinite** amount of **integers** (starting from **1**)
- **`halves()`** - generates the halves of those integers (each integer **/ 2**)
- **`take(n, seq)`** - takes the **first n** halves of those integers

***Note: Complete the functionality in the skeleton and submit it in the judge system***

## Examples

| Test Code | Output |
|---|---|
| `take = solution()[0]`<br>`halves = solution()[1]`<br>`print(take(5, halves()))` | `[0.5, 1.0, 1.5, 2.0, 2.5]` |

## 5. Fibonacci Generator

Create a generator function called **`fibonacci()`** that generates the **Fibonacci numbers** infinitely (**starting from 0**). Each Fibonacci number is created by the **sum** of the **current** number **with the previous.**

***Note: Submit only the function in the judge system***

## Examples

| Test Code | Output |
|---|---|

| | |
|---|---|
| <pre>generator = fibonacci()<br>for i in range(5):<br>    print(next(generator))</pre> | <pre>0<br>1<br>1<br>2<br>3</pre> |