TP 1: Page Rank

The difficult part was to implement the product between a vector and a sparse matrix.
I choose to represents such matrix by dictionnaries.

Then, it was just about applying the formula of Page Rank algorithm.

Another difficult part was to extract correctly the infomation present on the toyset, in order to create the graph.

For Beta=1 and esp=0.001 I obtained :

```
{'node 1': 0.28576151529947913,
  'node 2': 0.14273834228515625,
  'node 3': 0.14288075764973956,
  'node 4': 0.14300028483072916,
  'node 5': 0.14273834228515625,
  'node 6': 0.14288075764973956})
```

For Beta = 0.8, the jump were autorized, I obtained :

```
 {'node 1': 0.27144166191245994,
  'node 2': 0.14205073558667264,
  'node 3': 0.14688009681043457,
  'node 4': 0.15069667329332564,
  'node 5': 0.14205073558667264,
  'node 6': 0.14688009681043457})
```

It is logical that the node 1 loose importance in so far as we can jump with the probability 0.2 when we are on it.

Concerning the Wikipedia pages, I obtained :

```
 {'Assembly_language.html': 0.03924721305817514,
  'Binary_file.html': 0.004527633803277401,
  'Boolean_data_type.html': 0.00931262282786769,
  'Bytecode.html': 0.01392558988966497,
  'C++.html': 0.030436228699575397,
  'COBOL.html': 0.02343469619163733,
  'C_(programming_language).html': 0.04065222495276738,
  'Comparison_of_programming_languages.html': 0.015043802272147742,
  'Compiler.html': 0.033443171620915736,
  'Computer.html': 0.0220896294761528,
  'Computer_hardware.html': 0.022940202786461014,
  'Computer_memory.html': 0.015622789647479022,
  'Computer_program.html': 0.020267176138101382,
  'Computer_science.html': 0.02525192681909772,
  'Control_flow.html': 0.00975031562778569,
  'Data_(computing).html': 0.007541860560966966 5,
  'Data_type.html': 0.01922532815298363,
  'Database.html': 0.011228673044288302,
  'Dynamic_programming_language.html': 0.012146789950983778,
  'Executable.html': 0.012632100163216937,
  'Fortran.html': 0.02417875832683193,
  'GNU_Compiler_Collection.html': 0.01562877388288963,
  'High-level_programming_language.html': 0.022596278607702656,
  'Imperative_programming.html': 0.023105607719844153,
  'Instruction_set.html': 0.017393205084948432,
  'Integer_(computer_science).html': 0.009185803223652303,
  'JavaScript.html': 0.024491717448430445,
  'Java_(programming_language).html': 0.0335578160588575,
  'Kernel_(computing).html': 0.00458215434186174,
  'Lexical_scope.html': 0.0015637770689126325,
  'Linux.html': 0.012203973358538192,
  'Lisp_(programming_language).html': 0.02820834779985951,
  'List_of_programming_languages.html': 0.015001494219676144,
  'Logic_programming.html': 0.01274042866058705,
  'Low-level_programming_language.html': 0.015637332330421184,
  'Memory_address.html': 0.007767807524450869,
  'Method_(computer_programming).html': 0.006290642593192322,
  'Object-oriented_programming.html': 0.0290991846835297,
  'Object_(computer_science).html': 0.016062574604458212,
  'Operating_system.html': 0.03757216089586086,
  'Pointer_(computer_programming).html': 0.014857999162513767,
  'Porting.html': 0.00895398222064215,
  'Program_(machine).html': 0.0037725837261121135,
  'Programming_language.html': 0.048234437141462236,
  'Python_(programming_language).html': 0.026741833724812526,
  'Scripting_language.html': 0.01735443970022184,
  'Snowball_programming_language.html': 0.0008144992148488653,
  'Software.html': 0.013603491999745632,
  'Software_portability.html': 0.012785472843536233,
  'Source_code.html': 0.021408665445354306,
  'Strong_and_weak_typing.html': 0.004660149816214593,
  'Subroutine.html': 0.017025965064843145,
  'Type_system.html': 0.020361718055214053,
  'Unix.html': 0.019371535401587748,
  'Virtual_machine.html': 0.0244654123648392})
```

PS : it is more efficient to multiply all results by 100 to see the importance between 0 an 100.