Exercise 0: Python introduction

- Any questions?
Lecture last week: Boolean queries

- Grepping
- Inclusion, occurrence, order
- Difference between set/bag/list of terms
- Incidence matrix
  - Boolean queries
Lecture last week: Boolean queries

- Precision/Recall
Lecture last week: Boolean queries

- Inverted index
  - Memorize terminology
- Boolean queries
- Intersection & Union algorithms
- Optimizing
Exercise 1: Boolean Retrieval

- Theoretical questions on Moodle
  - True/false
  - Multiple choice
  - Algorithm design
- Struggles? Revise slides and/or book
Exercise 1: Boolean Retrieval

- Practical part: Jupyter notebooks
  - Implement simple inverted index for Boolean Retrieval

- Already implemented for you:
  - Boolean query parser
    - Handles error which can be caught using `ParseException`
  - Tokenizer
    - Handles tokenizing of documents and simple preprocessing of them
Exercise 1: Boolean Retrieval

- Your part:
  - Choose data structure

    ```python
    documents = dict()
    the_index = None
    ```

- Add a document and its terms to the inverted index

```python
documentid_counter = 1
def add_document(path):
    ...
    Add a document to the inverted index. Return the document's document ID.
    Remember the mapping from document ID to document in the `documents`
    data structure.
    ...
    # make sure that we access the global variables we have defined
    global the_index, documents, documentid_counter
    print("Adding \'s\' to index\" % path)
    pass
```

<table>
<thead>
<tr>
<th>Terms</th>
<th>Terms</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td>1, 4, 9, 10</td>
</tr>
<tr>
<td>u</td>
<td>5, 6, 7</td>
</tr>
<tr>
<td>v</td>
<td>2, 4, 6, 8, 10</td>
</tr>
<tr>
<td>w</td>
<td>5</td>
</tr>
<tr>
<td>x</td>
<td>1, 3, 4, 7</td>
</tr>
<tr>
<td>y</td>
<td>5, 8, 10</td>
</tr>
</tbody>
</table>
Exercise 1: Boolean Retrieval

- Your part:
  - Implement the intersection and union algorithm

```python
def intersect(p1, p2):
    """
    Method to compute the intersection of two postings lists. Takes
    postings lists as arguments and returns the intersection.
    """
    pass

def union(p1, p2):
    """
    Method to compute the union of two postings lists. Takes
    postings lists as arguments and returns the union.
    """
    pass
```

- Test it using provided examples
Exercise 1: Boolean Retrieval

Your part:
- Implement `execute_query(query)`
  - Remember the flattened structure of the query obtained using `process_ast`
  - Use `.op` to get operation of query
  - Use `.args` to get arguments of query.
    - Can be token or another query -> maybe use recursion
  - Maybe build subroutines handling e.g. negation
  - Maybe reuse or adapt `intersect` and `union` method from above
- Check with the examples below
Exercise 1: Boolean Retrieval

- To easy and boring for you?
  - Increase the complexity of queries your code is able to handle!
  - Try to further optimize the complexity of your algorithm