Stefano Mule – 4671228

Riccardo Cereghino – 4651066

```python
def selector(el: Dict[str, Any], mode: str, operators: List[Tuple[Callable, str,
Any]]) -> bool:
    """
    Given a List of comparisons (operators, ex: x < 3) returns a boolean expressing a
simple logic port for all
    operators, it is used in :func:`select`
    """
    result = False if mode == 'or' else True

    for op, k, v in operators:
        _result = op(el.get(k), v)

        if mode == 'or':
            result = _result or result
        else:
            result = _result and result
    return result
```

```python
def csv_reader(file_name: str) -> List[str]:
    """Generates an iterator per line from a file encoded in utf8, specified with
file_name"""
    lines = []
    for line in open(file_name, "r", encoding="utf8"):
        lines.append(line)
    return lines
```

```python
def team_goals(game: dict[str, Any], team_name: str):
    if game['home_team'] == team_name:
        return int(game['home_score'])
    return int(game['away_score'])
```

```python
def generate_games(file: str) -> Iterator[Dict[str, str]]:
    """
    Iterates through a csv file (path), picks the first line to be used
    as keys for the yielded list of returning dict
    """
    csv_gen = csv_reader(file)

    columns = row_splitter(csv_gen.pop(0))

    rows = []
    for row in csv_gen:
        rows.append(dict(zip(columns, row_splitter(row))))
    return rows
```

```python
def sum_goals(games: Iterator[dict[str, Any]], team_name: str) -> int:
    return reduce(lambda res, game: res + team_goals(game, team_name), games, 0)
```

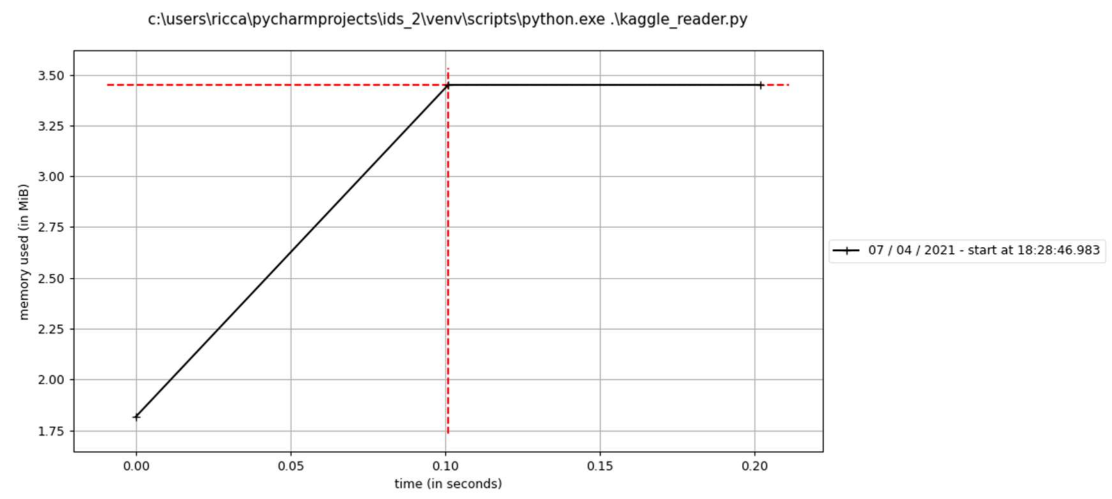
```python
def row_splitter(row: str) -> List[str]:
    """Given a string returns a csv row, splits the cells into elements of a list

    - input is in the form::
          row = "a,b,c\\n"
    """
    return row[:-1].split(',')
```

```python
def operators_reader(**kwargs: Union[str, Any]) -> List[Tuple[Callable, str, Any]]:
    """
    Given any number of kwargs in the form:
    - input is in the form::

        team_name__eq="Italy"
        avg_goals_scored__gte=1

    It returns a list of functions from the operator library, based on the *__eq*
section of the kwargs keyword.

    The arguments of the operator function will be, on the left, the value of the key
of the yielded dict and on the
    right the value od the key of the corresponding kwargs value.

    It is used in :func:`select`.
    """
    operators = []
    for kw in kwargs:
        if '__' in kw:
            _kw, op = kw.split('__')
            # only allow valid operators
            if op not in ('lt', 'le', 'eq', 'ne', 'ge', 'gt'):
                raise Exception("Invalid Operator")
        else:
            op = 'eq'
            _kw = kw

        _operator = getattr(operator, op)

        operators.append((_operator, _kw, kwargs[kw]))

    return operators
```

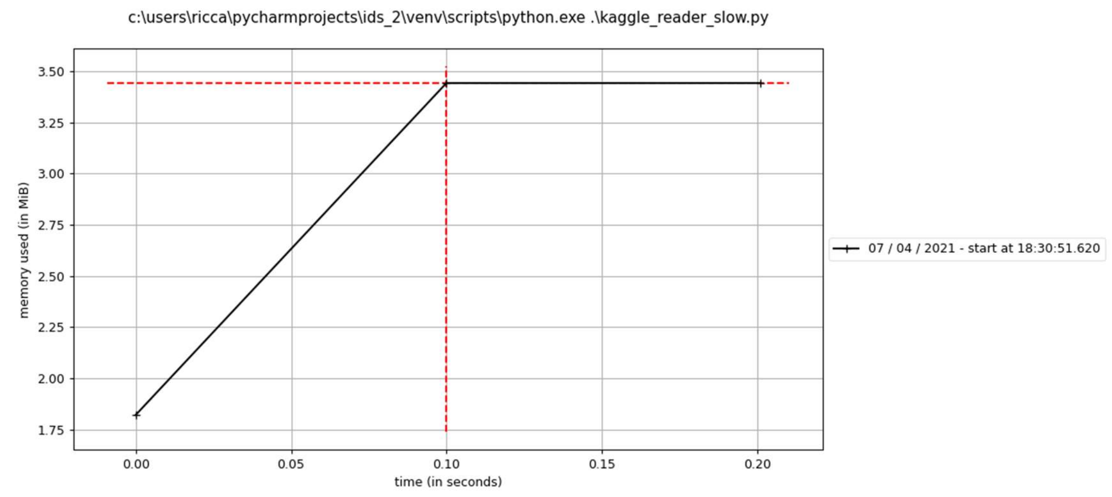
```python
def select(it: Iterator[Dict[str, Any]], **kwargs) -> Iterator[dict[str, Any]]:
    """
    Given a dict iterator (it) and any number of kwargs,
    - in the form::

        team_name__eq="Italy"
        avg_goals_scored__gte=1

    Returns a :func:`filter` iterator, filtering based on the condition specified in
kwargs`.
    """
    mode = kwargs.pop('mode', 'or')
    operators = operators_reader(**kwargs)

    elements = []

    for el in it:
        if selector(el, mode, operators):
            elements.append(el)

    return elements
```

Plots

Eseguendo il codice con mplot:

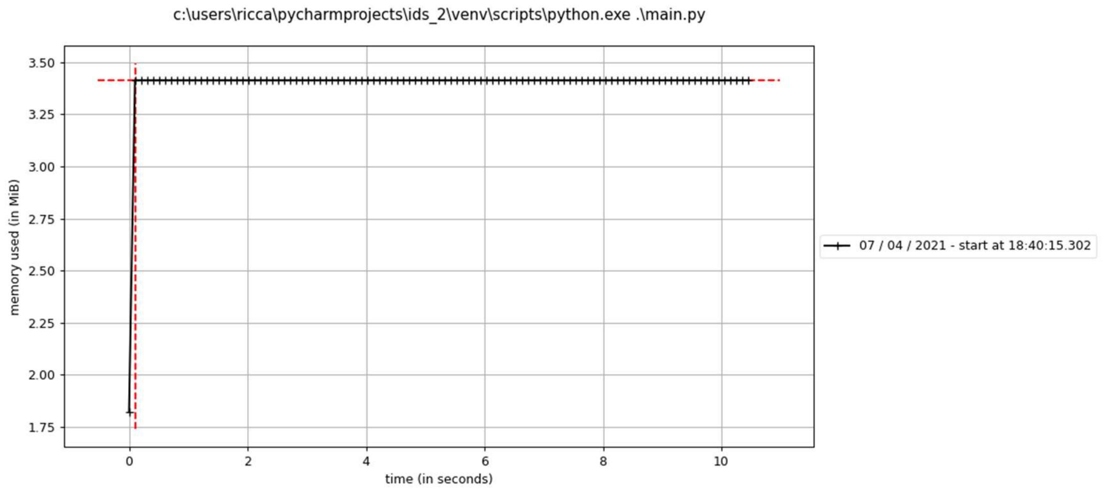


Modificando il codice per rimuovere gli yield in favore di ritorni di lista, non si rilevano cambiamenti, probabilmente dovuti al fatto che non ci sono abbastanza dati in memoria da appesantire il programma:



Invece utilizzando una versione del programma sviluppata utilizzando la libreria pandas ci sono notevoli differenze:

p


c:\users\ricca\pycharmprojects\ids_2\venv\scripts\python.exe .\main.py
memory used (in MiB)
time (in seconds)
07 / 04 / 2021 - start at 18:40:15.302
3.50
3.25
3.00
2.75
2.50
2.25
2.00
1.75
0
2
4
6
8
10