

Код программы

```
class Partition:
    '''Used to store partitions'''
    def __init__(self, _id, _name, _pages, _document_id):
        self.id = _id
        self.name = _name
        self.pages = _pages
        self.document_id = _document_id

class Document:
    '''Used to store documents'''
    def __init__(self, _id, _name):
        self.id = _id
        self.name = _name

class Link:
    '''Used to implement many-to-many relation'''
    def __init__(self, _part, _doc):
        self.part_id = _part.id
        self.doc_id = _doc.id

parts = [
    Partition(0, 'Правила', 20, 0),
    Partition(1, 'Разрешения', 30, 0),
    Partition(2, 'Ограничения', 12, 3),
    Partition(3, 'Определения', 8, 1),
    Partition(4, 'Вводная часть', 17, 2),
    Partition(5, 'Заключение', 14, 3)
]

docs = [
    Document(0, 'Правила поведения'),
    Document(1, 'Правила приема на работу'),
    Document(2, 'Правила перевода денежных средств'),
    Document(3, 'Ограничения на срок обучения')
]

links = [
    Link(parts[0], docs[0]),
    Link(parts[0], docs[1]),
    Link(parts[0], docs[2]),
    Link(parts[1], docs[0]),
    Link(parts[2], docs[3]),
    Link(parts[3], docs[0]),
    Link(parts[4], docs[1]),
    Link(parts[4], docs[2]),
    Link(parts[5], docs[1]),
    Link(parts[5], docs[3]),
]

def main():
    #implementation of one-to-many relation
    one_to_many = [
        (p.name, p.pages, d.name) #add name of doc and part and nr. of pages
        for p in parts           #loop over all parts
    ]
```

```

        for d in docs                #loop over all docs
        if d.id == p.document_id]    #take if p is part of d

print('Part 1')
#sort by nr. of pages
print(*sorted(one_to_many, key = lambda x: x[1]), sep='\n')

print(); print('Part 2')
#d.name may be not a key, so id is used
#id corresponds to name and count of pages
temp = {d.id:[d.name, 0] for d in docs}
for p in parts:                    #loop over parts
    #add count of pages to document containig this part
    temp[p.document_id][1] += p.pages
#keys are not used anymore, sort values by total amount of pages
print(*sorted(temp.values(), key=lambda x: x[1]), sep='\n')

print(); print('Part 3')
#same as previously
temp = {d.id:[d.name, []] for d in docs}
for p in parts:                    #loop over parts
    for l in links:                #loop over links
        if l.part_id == p.id:     #if part is linked via this record
            #add its name to the corresp. document
            temp[l.doc_id][1].append(p.name)
#leave only values containing 'Правила' in name
print(*filter(lambda x: 'Правила' in x[0], temp.values()), sep='\n')

if __name__ == '__main__':
    main()

```

## Результаты работы:

### Part 1

('Определения', 8, 'Правила приема на работу')  
('Ограничения', 12, 'Ограничения на срок обучения')  
('Заключение', 14, 'Ограничения на срок обучения')  
('Вводная часть', 17, 'Правила перевода денежных средств')  
('Правила', 20, 'Правила поведения')  
('Разрешения', 30, 'Правила поведения')

### Part 2

['Правила приема на работу', 8]  
['Правила перевода денежных средств', 17]  
['Ограничения на срок обучения', 26]  
['Правила поведения', 50]

### Part 3

['Правила поведения', ['Правила', 'Разрешения', 'Определения']]  
['Правила приема на работу', ['Правила', 'Вводная часть', 'Заключение']]  
['Правила перевода денежных средств', ['Правила', 'Вводная часть']]