```python
from operator import itemgetter


class PL:
    def __init__(self, pl_id, name):
        self.pl_id = pl_id
        self.name = name


class Syntax:
    def __init__(self, syn_id, name, pl_id, usefulness_grade=1.0):
        self.syn_id = syn_id
        self.name = name
        self.pl_id = pl_id
        self.usefulness_grade = usefulness_grade


class PL_Syn:
    def __init__(self, pl_id, syn_id):
        self.pl_id = pl_id
        self.syn_id = syn_id


PLs: list[PL] = [
    PL(1, 'Python'),
    PL(2, 'C++'),
    PL(3, 'C#'),
    PL(4, 'Java')
]
Syntaxes = [
    Syntax(1, 'do-while', 2, 0.5),
    Syntax(2, 'pointer', 2, 2.0),
    Syntax(3, 'interface', 3, 2.0),
    Syntax(4, 'for', 1, 1.5),
    Syntax(5, 'class', 4, 2.0)
    # Оценки полезности проставлены случайным образом и не отражают позицию
автора
]
PL_Syns = [
    PL_Syn(1, 1),
    PL_Syn(1, 4),
    PL_Syn(1, 5),
    PL_Syn(2, 1),
    PL_Syn(2, 2),
    PL_Syn(2, 4),
    PL_Syn(2, 5),
    PL_Syn(3, 1),
    PL_Syn(3, 3),
    PL_Syn(3, 4),
    PL_Syn(3, 5),
    PL_Syn(4, 1),
    PL_Syn(4, 4),
    PL_Syn(4, 5)
]


def request1():
    one_to_many = [(syn.name, pl.name)
                   for pl in PLs
                   for syn in Syntaxes
                   if (pl.pl_id == syn.pl_id)]
    result = sorted(one_to_many, key=itemgetter(1))
    print(*result, sep="\n")
```

```python
def request2():
    one_to_many = [(syn.name, syn.usefulness_grade, pl.name)
                   for pl in PLs
                   for syn in Syntaxes
                   if (pl.pl_id == syn.pl_id)]
    result = list()
    for pl in PLs:
        syns = list(filter(lambda i: i[2] == pl.name, one_to_many))
        if len(syns) != 0:
            sum_ug = sum(ug for _, ug, _ in syns)
            result.append((pl.name, sum_ug))
    result = sorted(result, key=itemgetter(1), reverse=True)
    print(*result, sep="\n")


def request3():
    many_to_many_temp = [(pl.name, p_s.syn_id)
                         for pl in PLs
                         for p_s in PL_Syns
                         if p_s.pl_id == pl.pl_id]
    many_to_many = [(pl_name, syn.name)
                    for pl_name, syn_id in many_to_many_temp
                    for syn in Syntaxes
                    if syn.syn_id == syn_id]

    result = dict()
    for pl in PLs:
        if "C" in pl.name:
            filtered = list(filter(lambda i: i[0] == pl.name, many_to_many))
            s_names = [i for _, i in filtered]
            result[pl.name] = s_names
    print(result, sep="\n")


def main():
    request1()
    print("\n")
    request2()
    print("\n")
    request3()


if __name__ == "__main__":
    main()
```

Результаты выполнения:

Задание А1:

```
('interface', 'C#')

('do-while', 'C++')

('pointer', 'C++')

('class', 'Java')

('for', 'Python')
```

Задание А2:

```
('C++', 2.5)

('C#', 2.0)

('Java', 2.0)

('Python', 1.5)
```

Задание А3:

```
{'C++': ['do-while', 'pointer', 'for', 'class'], 'C#': ['do-while',
'interface', 'for', 'class']}
```

```
('interface', 'C#')
('do-while', 'C++')
('pointer', 'C++')
('class', 'Java')
('for', 'Python')


('C++', 2.5)
('C#', 2.0)
('Java', 2.0)
('Python', 1.5)


{'C++': ['do-while', 'pointer', 'for', 'class'], 'C#': ['do-while', 'interface', 'for', 'class']}
```