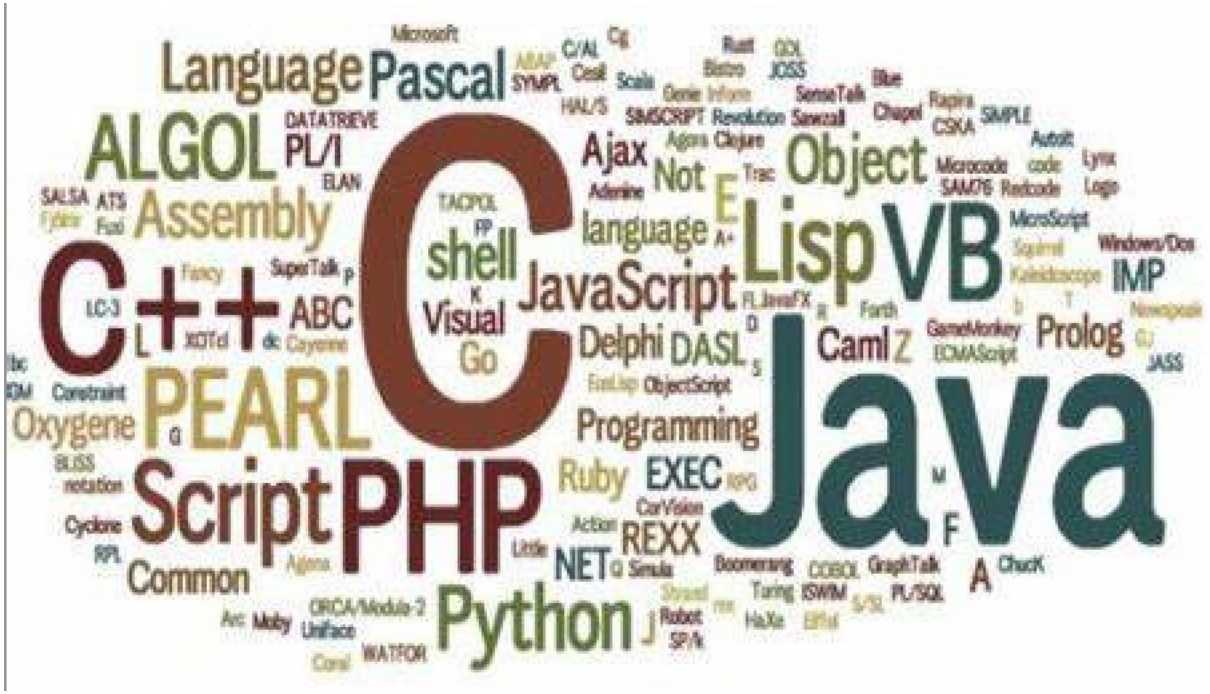

面向对象技术

13) 2022-05-06

14) 2022-05-13

15) 2022-05-27

刘聪

本节课

- 项目3: 中国象棋对弈程序
  - https://gitee.com/cse-oop/project-3
  - 学习目的: 学习一个包含复杂界面，控制逻辑，运行规则，和对弈搜索算法的OOP设计例子。
  - 扩展该项目的同学请fork该repo
  - 扩展该项目的方向
    - https://blog.icytown.com/algo/alpha-beta-pruning/
    - 深度学习: AlphaZero, AlphaMu …
- 课程安排
  - 象棋规则的简介
    - https://zhuanlan.zhihu.com/p/379717618
  - 双人象棋对弈界面，下棋的控制逻辑，实现象棋运行规则
    - chess.py
  - 人机对弈搜索算法，使用远程调用把计算放到服务器端
    - auto_chess.py
    - ajax.py

同时利用Javascript和Python的资源

- Javascript
  - DOM javascript.document
  - javascript 全局函数和类
    - javascript.console.log(“hello”)
    - date = javascript.Date.new()
    - hour = date.getHour().data()
- Python
  - 使用纯Python编写的库文件
    - https://github.com/micropython/micropython-lib
    - 稍作更改以适应我们的简化编译器
  - 例子
    - heapq.py
    - 简化自 https://github.com/micropython/micropython-lib/blob/master/python-stdlib/heapq/heapq.py

人机对弈搜索算法

- Alpha-beta 搜索树
  - 简单来说：
    - 如何决定下一步怎么走？
    - 穷尽下一步后的所有棋盘，计算各个棋盘的分数，选择最高分的下一步。
    - 如何计算每个棋盘的分数？
      - 粗略计算(基础步)，所有棋子的加权和
      - 精确计算(递归步)：
        - 一个棋盘的分数=下一步(对手)棋盘的最大分数的负数
        - 直观解释:
          - 对手会选能最大化他的分数的下一步
          - 在对手看来越好的分数，在我看来越差
  - Alpha-beta 剪枝 (待实现)
    - 对于不看好(分数低)的棋盘不再去深思(搜索)

把计算放到服务器上

- 人机对弈搜索函数
  - move = auto_move(self.chess_board)
  - 在网页上运行非常慢
    - Javascript 本身就很慢，再用Javascript解释Python更慢
    - 我们的 Python运行环境主要优化的是程序加载时间，对运行速度的优化放在第二位
  - 我们将在服务器中调用 auto_move
    - 可以直接调用
      - 我们的前后端都是 Python, 可以共享代码
      - 相当与把 auto_move挪到后端，只不过代码仍然留在前端的文件夹里。
    - 但是前端需要把当前棋盘发给后端，后端需要把计算结果返回给前端
      - 我们使用 AJAX (**A**synchronous **J**avaScript **A**nd **X**ML) 在前后端之间传送 JSON (JavaScript Object Notation) 数据
      - 我们再进一步把 AJAX调用包装为远程调用
        - 感觉起来就是网页上可以直接调用服务器中的函数
        - 具体实现细节请自己查看 ajax.py

把计算放到服务器上

- 首先在后端添加可远程调用的函数
  - 后端 __main__.py

```
48 # 登记函数 RPC auto_move
49 rpc_registry['rpc_auto_move'] = rpc_auto_move
```

  - 那么函数后端的 rpc_auto_move 函数就可以在前端使用名字 'rpc_auto_move'调用了
- 前端改动
  - 注意: 远程调用的函数的参数和返回值需要能变为JSON
  - 在前端 auto_chess.py 把

```
move = auto_move(self.chess_board)
```

  - 改为远程调用 RPC (remote procedure call)
    - 远程函数前加上 ajax.rpc.

```
33          from . import ajax
34          board_key = _board_key(self.chess_board) # board_key 可变为 JSON
35          move = ajax.rpc.rpc_auto_move(board_key)
```

把计算放到服务器上

- 最后看看后端怎么调用 auto_move
  - 需要先把能转化为 JSON 的 board_key 变回 auto_move 所需的对象
    - 使用 _board_from_key

```
43 def rpc_auto_move(board_key):
44     from web.py_lib import auto_chess
45     board = auto_chess._board_from_key(board_key)
46     return auto_chess.auto_move(board)
```