纯函数：

笔记本：	1.有关印象笔记		
创建时间：	2020/6/21 15:49	更新时间：	2020/7/19 14:08
作者：	1639079350@qq.com		
标签：	3.纯函数/lodash		

纯函数：

- 输入相同相同就输出，而且没有副作用。（类似于数学中的函数，用来描述输入和输出之间的关系）。
- slice：纯函数；splice：不纯的函数。

lodash： https://www.lodashjs.com/
1.一个纯函数的功能库，提供了一些方法。
2. 和数组的有些方法不同，如reverse有参数：
3.好处：
（1）可缓存：_.memoize：有相同的输出，可以缓存结果；【例子只执行一次，剩下的都是从缓存中获取，可提高性能。】
（2）可测试：有输入输出；
（3）并行处理：多线程的并行操作共享的内存数据可能出现意外。纯函数不访问共享的内存数据，在并行环境下可任意运行纯函数。
4.副作用：当函数依赖于外部的状态就无法保证相同的输入有相同的输出了，就会带来副作用；
（1）副作用的来源：配置文件/数据库/获取用户的输入。
（2）副作用不可能完全禁止，尽可能控制在可控制的范围内。

```
const { log} = console;
const arr = ['jack','tom','lucy','kate'];
log(_.first(arr)); // 'jack'
log(_.last(arr));  // 'kate'
log(_.toUpper(_.first(arr)));  // 'JACK'
log(_.reverse(arr));  // ['kate','lucy','tom','jack'];


// 缓存
function getArea(r) {
    log(r); // 4(只输出一遍)
    return Math.PI * r * r;
}
let getAreaWithMemory = _.memoize(getArea);
log(getAreaWithMemory(4)); // 50.26548245743669
```

```
log(getAreaWithMemory(4)); // 50.26548245743669
log(getAreaWithMemory(4)); // 50.26548245743669

// 模拟_.memorize函数的实现
function memoize(f) {
    let cache = {};
    return function () {
        let key = JSON.stringify(arguments);
        cache[key] = cache[key] || f.apply(f,arguments);
        return cache[key];
    }
}
let AreaWithMemory = memoize(getArea);
log(AreaWithMemory(4));

// 副作用
// 不纯
let mini = 18;
function checkAge( age) {
    return age >= mini;
}
// 纯函数（硬编码，后期可以通过柯里化解决）
function checkAge( age) {
    let mini = 18;
    return age >= mini;
}
```