Function

Python 基础入门

回顾

字典 和 对应

Key 和 Value

字典 和 列表

in 查找

不存在的 Key

使用字典 加密

思考

生活中的 事情

很多 事情，往往包括多个 步骤，例如 榨果汁

- ✓ 插上榨汁机的插头
- ✓ 打开榨汁机的盖子
- ✓ 向榨汁机中放入水果
- ✓ 打开开关，开始榨汁
- ✓ 关闭开关，停止榨汁
- ✓ 倒出榨好的果汁
- ✓ 清洗榨汁机
- ✓ 拔下榨汁机的插头

```
# 插上榨汁机的插头
# 打开榨汁机的盖子
# 向榨汁机中放入水果
# 打开开关，开始榨汁
# 关闭开关，停止榨汁
# 倒出榨好的果汁
# 清洗榨汁机
# 拔下榨汁机的插头
```

思考

这些事情 经常要做

如果 每天 都要榨一杯果汁

```
# 插上榨汁机的插头
# 打开榨汁机的盖子
# 向榨汁机中放入水果
# 打开开关，开始榨汁
# 关闭开关，停止榨汁
# 倒出榨好的果汁
# 清洗榨汁机
# 拔下榨汁机的插头
```


```
# 插上榨汁机的插头
# 打开榨汁机的盖子
# 向榨汁机中放入水果
# 打开开关，开始榨汁
# 关闭开关，停止榨汁
# 倒出榨好的果汁
# 清洗榨汁机
# 拔下榨汁机的插头
```

```
# 插上榨汁机的插头
# 打开榨汁机的盖子
# 向榨汁机中放入水果
# 打开开关，开始榨汁
# 关闭开关，停止榨汁
# 倒出榨好的果汁
# 清洗榨汁机
# 拔下榨汁机的插头
```

```
# 插上榨汁机的插头
# 打开榨汁机的盖子
# 向榨汁机中放入水果
# 打开开关，开始榨汁
# 关闭开关，停止榨汁
# 倒出榨好的果汁
# 清洗榨汁机
# 拔下榨汁机的插头
```

思考

更 省力 的方法

- ✓ 找个 老婆 / 老公
- ✓ 教会她 / 他 详细的流程
- ✓ 每天和她 / 他说，去 榨果汁
- ✓ 然后就得到了 一杯果汁

函数

用 Python 编写 函数

- ✓ 用 函数 来实现一项 功能
- ✓ 把实现的 细节代码 都写在函数里
- ✓ 使用的时候 调用函数 即可

```
def get_juice():
    print('=' * 20)
    print('打开榨汁机')
    print('放入水果')
    print('关闭榨汁机')
    print('=' * 20)

get_juice()
get_juice()
get_juice()
```

函数

函数的 声明

- ✓ 使用 def 声明一个函数
- ✓ 给函数取一个 名称
- ✓ 命名规范和 变量 类似
- ✓ 使用 小括号、 冒号 和 缩进
- ✓ 编写函数需要完成的 代码

```
def get_juice():
    print('=' * 20)
    print('打开榨汁机')
    print('放入水果')
    print('关闭榨汁机')
    print('=' * 20)
```

函数

函数的 使用

- ✓ 完成了函数的 声明 之后
- ✓ 使用函数的 名称 即可 调用
- ✓ 一次声明，随意使用，一劳永逸

```
def get_juice():
    print('=' * 20)
    print('打开榨汁机')
    print('放入水果')
    print('关闭榨汁机')
    print('=' * 20)

get_juice()
get_juice()
get_juice()
```

参数

让函数更加 灵活多样

- ✓ 今天想喝 苹果汁，明天想喝 西瓜汁
- ✓ 在声明部分使用 参数
- ✓ 不同的 参数，对应不同的 运行结果

形参：声明时使用，仅起 语义作用

实参：调用时使用，必须是 具体的值

形参

```
def get_juice(fruit):
    print('=' * 20)
    print('打开榨汁机')
    print('放入', fruit)
    print('关闭榨汁机')
    print('=' * 20)

get_juice('Apple')
get_juice('Watermelon')
```

实参

练习

1 累加和

- ✓ 编写一个函数，计算 1 到 N 之间所有 正整数 的和
- ✓ N 也是一个 正整数
- ✓ 调用 写好的函数，N 分别为 5、10、15

```
def cum_sum(N):
    # your codes

cum_sum(5)
cum_sum(10)
cum_sum(15)
```

返回值

让函数 返回 结果

- ✓ 果汁榨好之后，记得 端给我
- ✓ 让函数返回结果，用于后续 进一步使用
- ✓ 返回的值，可以是 任意类型
- ✓ 使用 return 返回结果
- ✓ return 代表函数的 结束

```
def cum_sum(N):
    s = 0
    for i in range(N):
        s += i + 1
    return s

s = cum_sum(100)
print(s)
```

返回值

回忆 一下

学过的函数中，哪些 有返回值，哪些 没有返回值

变量、类型、条件、循环、列表、字符串、字典

作用域

函数的 作用域

- ✓ 声明中的变量，仅在函数内部 有效
- ✓ 函数调用时 临时产生
- ✓ 函数结束后 马上销毁

年年岁岁花相似，岁岁年年人不同

```
def cum_sum(N):
    s = 0
    for i in range(N):
        s += i + 1
    return s

print(cum_sum(100))
print(N)
print(s)
print(i)
```

练习

2 是 质数 吗

- ✓ 编写一个函数，判断 N 是否为 质数
- ✓ N 是一个 正整数
- ✓ 返回 True 或 False
- ✓ 调用 写好的函数，N 分别为 1、19、1234567

```
n = 19
flag = True
for i in range(2, n):
    if n % i == 0:
        flag = False
        break
if flag:
    print(n, '是质数')
else:
    print(n, '不是质数')
```

```
def is_prime(n):
    # your codes
```

练习

3 很多 质数

- ✓ 编写一个函数，找到所有不超过 N 的 质数，N 是一个 正整数
- ✓ 将符合要求的数存到 列表 中并 返回
- ✓ 调用 写好的函数，N 分别为 1、20、100

```
def get_primes(N):
    result = []
    if N > 1:
        for i in range(2, N + 1):
            # your codes
    return result
```

作业

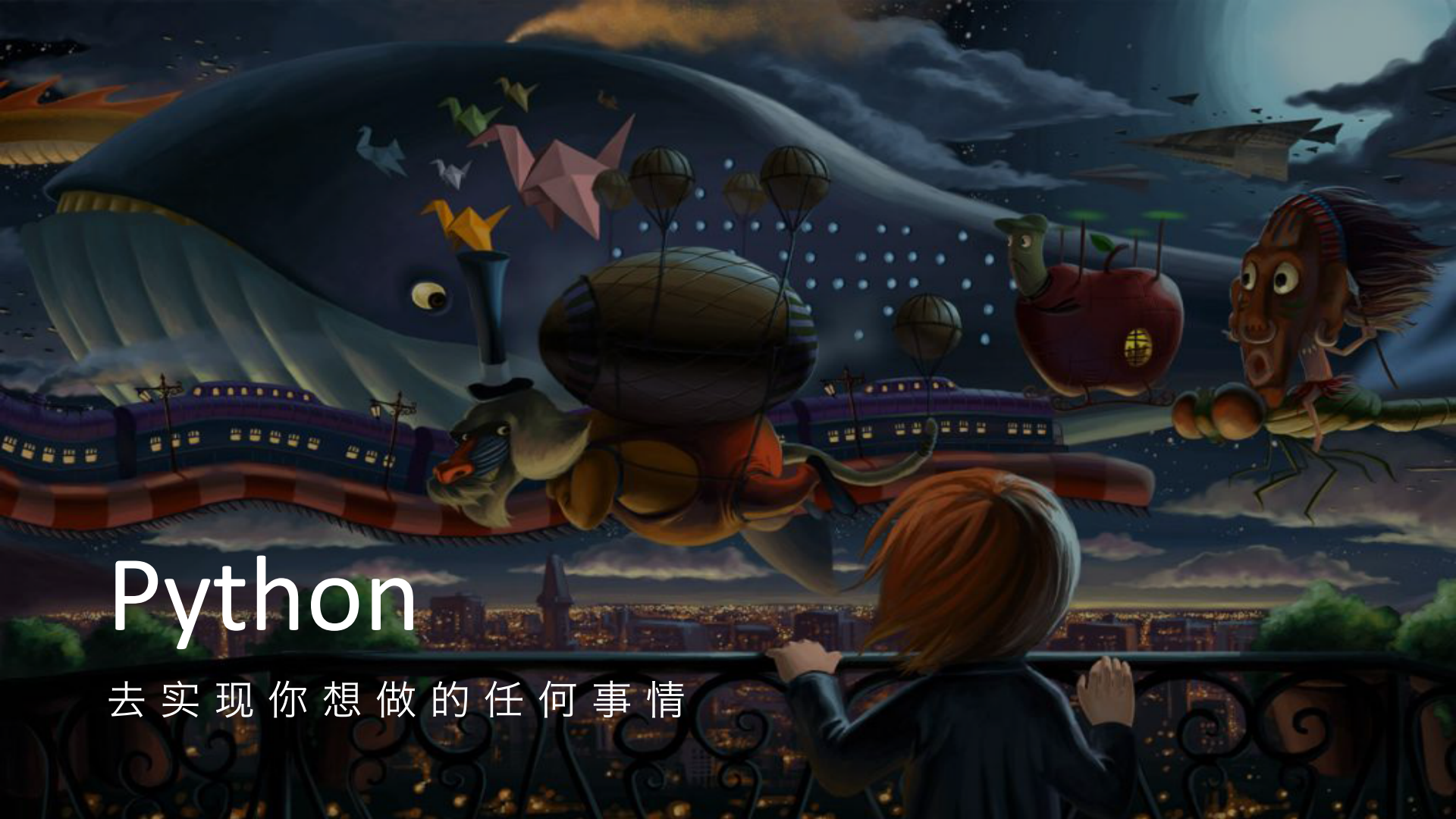
1 矩形 计算

- ✓ 编写一个函数，参数为矩形的 长 和 宽
- ✓ 计算矩形的 面积 和 周长
- ✓ 返回计算的 两个结果
- ✓ 调用 写好的函数

作业

2 字符串 查删

- ✓ 编写一个函数，共 3 个参数，分别为：
  - ✓ 一个 长的 字符串
  - ✓ 一个 短的 字符串
  - ✓ 一个 布尔值
- ✓ 在长串中 查找并删除 短串，返回处理后的结果
- ✓ 第三个参数为 True 则 全部 删除，为 False 则 至多 删除一次
- ✓ 调用写好的函数，尝试 不同的情况


Python
去 实 现 你 想 做 的 任 何 事 情