

# CTF Class 2: Python实践

---

**Python for Fresh CTFers**

# Topic 1: Calculator

任务一：尝试搭建一个简易的计算器

我想要实现开方运算…

但是我好像我不知道怎么实现开方运算… / 实现很复杂我不太想亲自实现…

# Topic 1: Calculator

任务一：尝试搭建一个简易的计算器

我想要实现开方运算…

但是我好像我不知道怎么实现开方运算… / 实现很复杂我不太想亲自实现…

我们可以使用他人已经写好的函数！（或其他东西）

库 —— 库函数

```
from math import sqrt
```

练习：将开平方功能添加到先前写好的计算器中。

① 读取输入 — ② 进行计算 — ③ 将计算得到的值进行打印

```
import 库名  
from 库名 import 函数名  
import 库名 as 自定义库名  
from 库名 import 函数名 as 自定义函数名
```

Representer: Sike

# Topic 1: Calculator

任务一：尝试搭建一个简易的计算器

库 —— 库函数

```
from math import sqrt
```

```
a = 3
```

```
print(sqrt(a))
```

```
import 库名  
from 库名 import 函数名  
import 库名 as 自定义库名  
from 库名 import 函数名 as 自定义函数名
```

练习：将开平方功能添加到先前写好的计算器中。

① 读取输入 — ② 进行计算 — ③ 将计算得到的值进行打印

Representer: Sike

# Topic 1: Calculator

任务一：尝试搭建一个简易的计算器

库 —— 库函数

```
from math import sqrt  
  
a = 3  
print(sqrt(a))
```

练习：将开平方功能添加到先前写好的计算器中。

```
import math  
  
a = 3  
print(math.sqrt(a))
```

点号操作符

from 库名 import 函数名

函数名直接调用

import 库名

函数调用：

库名.函数名(函数参数)

# Topic 1: Calculator

任务一：尝试搭建一个简易的计算器

库 —— 库函数

```
from math import sqrt as sr  
  
a = 3  
print(sr(a))
```

from 库名 import 函数名 as 自定义函数名  
自定义函数名直接调用

import 库名 as 自定义库名  
函数调用：  
自定义库名.函数名(函数参数)

```
import math as mt  
  
a = 3  
print(mt.sqrt(a))
```

练习：将开平方功能添加到先前写好的计算器中。

# Topic 1: Calculator

## 任务一：尝试搭建一个简易的计算器

- 使用 pip 进行库安装
  - (python -m) pip install 库名称
  - 基于 requirement.txt 进行安装（自行检索）
- pip 的更新
  - python -m pip install --upgrade pip

`from 库名 import 函数名 as 自定义函数名`  
自定义函数名直接调用

`import 库名 as 自定义库名`  
函数调用：  
`自定义库名.函数名(函数参数)`

# Review & Feedback

- 计算机计算精度的问题
- `for`循环理解

```
for i in range(21, 1, -2):  
    print(f"{i + 1}", end = " ")
```

- 设计一个简单的交互程序：Read-Eval-Print Loop



# Topic 2: Classical Crypto~

## 任务二：一个简单的加解密系统

### Ex6: 凯撒密码

凯撒密码是一种替换加密的技术，明文中的所有字母都在字母表上向后（或向前）按照一个固定数目进行偏移后被替换成密文。如当偏移（密钥）为向后取2时，明文”abcd”被加密为”cdef”。



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

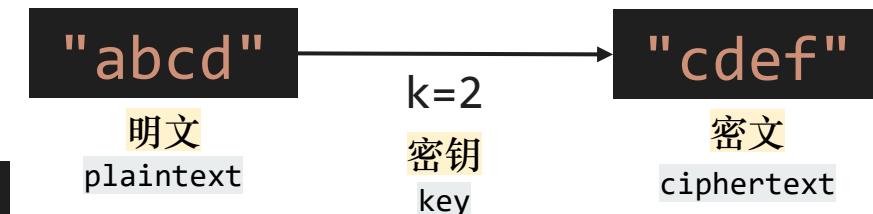
```
def encrypt_1(s: str, k: int = 3) -> str:
    result = []
    for c in s:
        # 处理大写字母
        if c.isupper():
            new_ord = (ord(c) - ord('A') + k) % 26 + ord('A')
            result.append(chr(new_ord))
        # 处理小写字母
        elif c.islower():
            new_ord = (ord(c) - ord('a') + k) % 26 + ord('a')
            result.append(chr(new_ord))
        # 非字母字符（数字、符号等）直接保留
        else:
            result.append(c)
    return ''.join(result)
```



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

```
def encrypt_1(s: str, k: int = 3) -> str:
    result = []
    for c in s:
        # 处理大写字母
        if c.isupper():
            new_ord = (ord(c) - ord('A') + k) % 26 + ord('A')
            result.append(chr(new_ord))
        # 处理小写字母
        elif c.islower():
            new_ord = (ord(c) - ord('a') + k) % 26 + ord('a')
            result.append(chr(new_ord))
        # 非字母字符（数字、符号等）直接保留
        else:
            result.append(c)
    return ''.join(result)
```



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

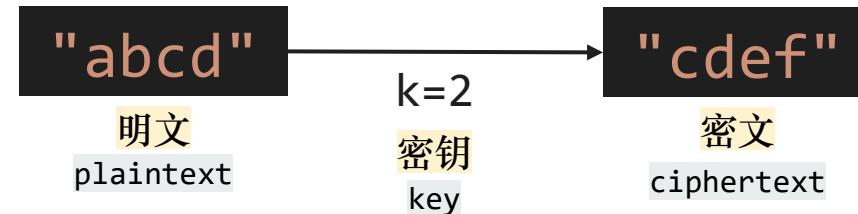


```
def encrypt_2(s: str, k: int = 3) -> str:
    result = []
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    for c in s:
        # 处理大写字母
        if c.isupper(): result.append(sub(c, 'A', k))
        # 处理小写字母
        elif c.islower(): result.append(sub(c, 'a', k))
        # 非字母字符（数字、符号等）直接保留
        else: result.append(c)
    return ''.join(result)
```

函数名 = lambda 变量: 表达式

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



函数名 = lambda 变量: 表达式

```
ADD = lambda x, y: x + y
DBL = lambda x: x ** 2
li = [1, 2, 3, 4, 5, 6]
```

练习：使用lambda表达式和reduce实现递归加/递归乘。

练习：实现阶乘（结合列表推导式）。

lambda表达式（匿名函数）可以很好地和list的一些操作结合使用。

map()

```
DBL_li = list(map(DBL, li)); print(DBL_li)
DBL_li_ = list(map(lambda x: x ** 2, DBL_li)); print(DBL_li_)
```

filter()

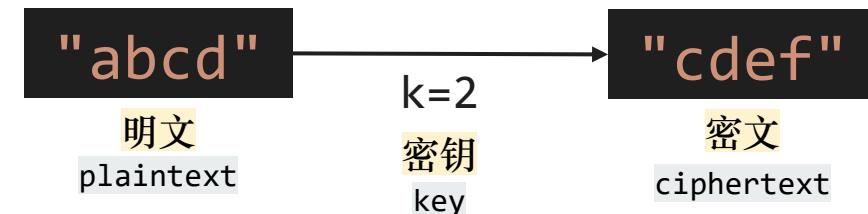
```
f_DBL_li = list(filter(DBL, li)); print(f_DBL_li)
f_DBL_li_ = list(filter(lambda x: x % 2 == 0, li)); print(f_DBL_li_)
```

reduce()

```
from functools import reduce
r_ADD_li = reduce(ADD, li); print(r_ADD_li)
r_ADD_li_ = reduce(lambda x, y: x + y, li); print(r_ADD_li_)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_2(s: str, k: int = 3) -> str:  
    result = []  
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))  
    for c in s:  
        # 处理大写字母  
        if c.isupper(): result.append(sub(c, 'A', k))  
        # 处理小写字母  
        elif c.islower(): result.append(sub(c, 'a', k))  
        # 非字母字符（数字、符号等）直接保留  
        else: result.append(c)  
    return ''.join(result)
```

# Topic 2: Classical Crypto~

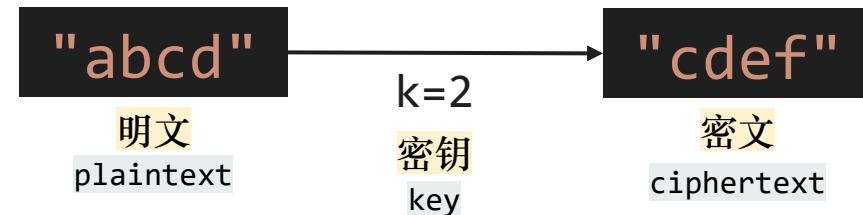
任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_2(s: str, k: int = 3) -> str:  
    result = []  
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))  
    for c in s:  
        # 处理大写字母  
        if c.isupper(): result.append(sub(c, 'A', k))  
        # 处理小写字母  
        elif c.islower(): result.append(sub(c, 'a', k))  
        # 非字母字符（数字、符号等）直接保留  
        else: result.append(c)  
    return ''.join(result)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_3(s: str, k: int = 3) -> str:
    result = []
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    for c in s:
        result.append(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c)
    return ''.join(result)
```

值1 if 条件 else 值2

若条件成立（为真）则取值1，否则取值2

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

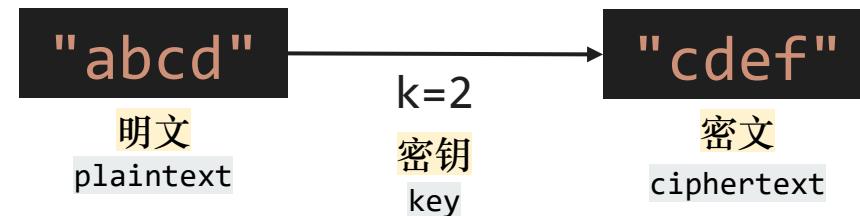
值1 if 条件 else 值2

若条件成立（为真）则取值1，否则取值2



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



值1 if 条件 else 值2

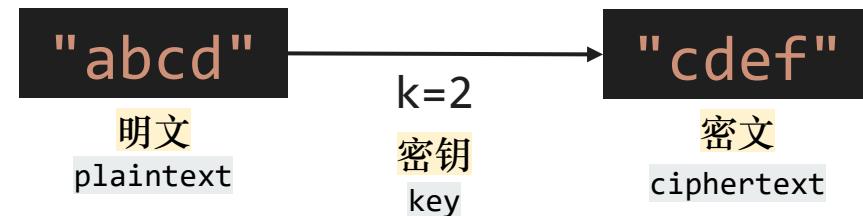
若条件成立（为真）则取值1，否则取值2

二元决策图与INF (ITE范式, If-then-else Normal Form)

```
expr = sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



值1 if 条件 else 值2

若条件成立（为真）则取值1，否则取值2

二元决策图与INF (ITE范式, If-then-else Normal Form)

```
expr = sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c
```

c.isupper()?

True

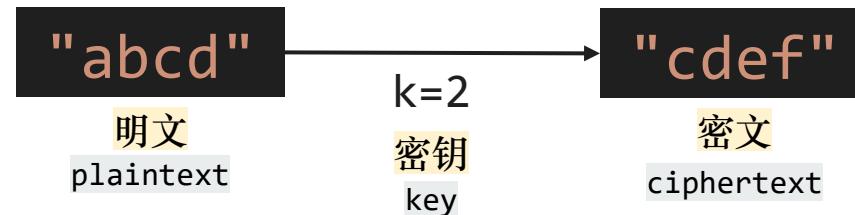
```
sub(c, 'A', k)
```

False

```
sub(c, 'a', k) if c.islower() else c
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

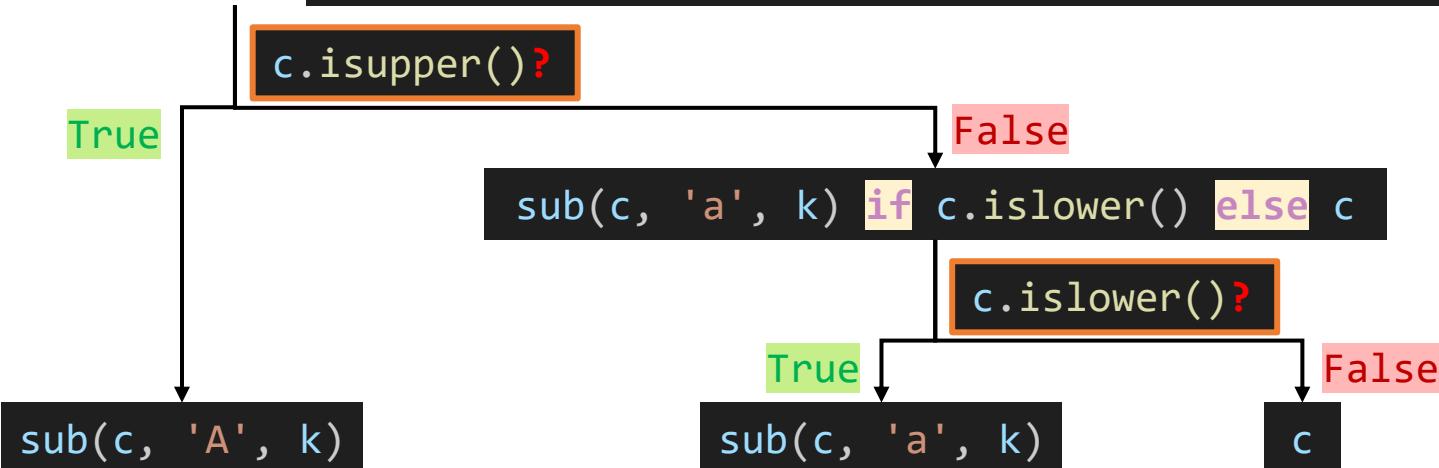


值1 if 条件 else 值2

若条件成立（为真）则取值1，否则取值2

二元决策图与INF (ITE范式, If-then-else Normal Form)

```
expr = sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c
```



注：任何一个条件表达式（布尔函数）都可以香农展开（Shannon's Expansion），从而可以递归地将其完全使用if-then-else的方式写出来。

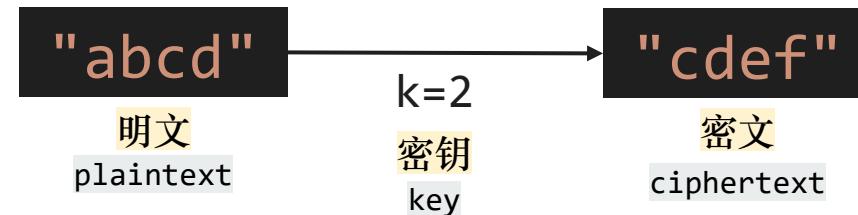
注：Python 3.14 添加了 `match` 功能。

练习：分段函数的简洁实现。对门数进行探究。

Representer: Sike

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



值1 if 条件 else 值2

若条件成立（为真）则取值1，否则取值2

二元决策图与INF (ITE范式, If-then-else Normal Form)

```
func1 = lambda x: x if x > 0 else -x

func2 = lambda x: x if x < -1 else -x if x < 1 else x
func2_ = lambda x: -x if -1 <= x < 1 else x

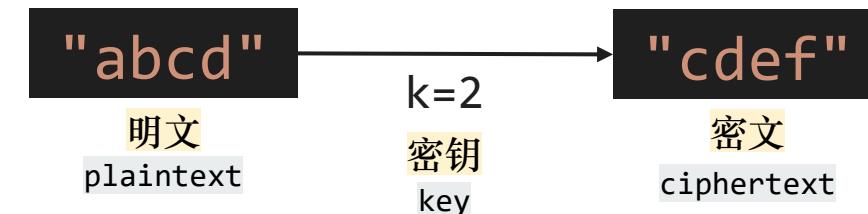
func3 = lambda x: "D" if x < 60 else ("C" if x < 80 else ("B" if x < 90 else "A"))
func3 = lambda x: ("D" if x < 60 else "C") if x < 80 else ("B" if x < 90 else "A")
func3 = lambda x: ((("D" if x < 60 else "C") if x < 80 else "B") if x < 90 else "A")
```

练习：分段函数的简洁实现。对门数进行探究。

Representer: Sike

# Topic 2: Classical Crypto~

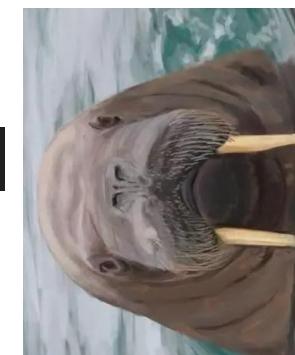
任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_3(s: str, k: int = 3) -> str:
    result = []
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    for c in s:
        result.append(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c)
    return ''.join(result)
```

补充：一个更加Tricky的写法（海象运算符：行内赋值）

```
sub = lambda ch, base, k: chr((ord(ch) - (o := ord(base)) + k) % 26 + o)
```



Representer: Sike

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_3(s: str, k: int = 3) -> str:
    result = []
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    for c in s:
        result.append(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c)
    return ''.join(result)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_4(s: str, k: int = 3) -> str:  
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))  
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

列表推导式

[表达式 for 变量 in 迭代器]

练习：生成一列表平方数

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

列表推导式

[表达式 for 变量 in 迭代器]

练习：生成一列表平方数

列表推导式+列表操作

练习：阶乘函数的实现

all()

```
my_isPrime = lambda n: all(n % i != 0 for i in range(2, int(n**0.5) + 1))
```

any()

```
hasPrime = lambda li: any(my_isPrime(i) for i in li)
```

sum()

```
primeCount = lambda li: sum(my_isPrime(i) for i in li)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

列表推导式

[表达式 **for** 变量 **in** 迭代器]

练习：生成一列表平方数

```
my_isPrime = lambda n: all(n % i != 0 for i in range(2, int(n**(1/2)) + 1))
```

列表推导式+列表操作

```
hasPrime = lambda li: any(my_isPrime(i) for i in li)
```

练习：阶乘函数的实现

```
primeCount = lambda li: sum(my_isPrime(i) for i in li)
```

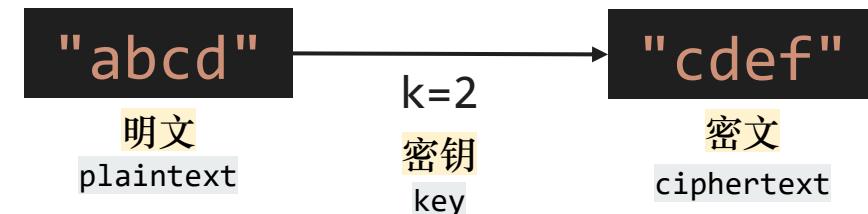
带条件列表推导式

[表达式 **for** 变量 **in** 迭代器 **if** 条件]

练习：生成小于N的素数列表

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

列表推导式

[表达式 **for** 变量 **in** 迭代器]

凡是这种一列数据的类型（数据结构），

元组推导式

(表达式 **for** 变量 **in** 迭代器)

一般都可以使用推导式。

集合推导式

{表达式 **for** 变量 **in** 迭代器}

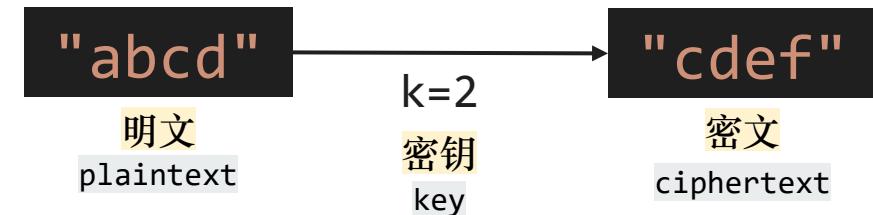
字典推导式

{键表达式 : 值表达式 **for** 变量 **in** 迭代器}

补充：元组推导式返回的是一个生成器表达式。（`next()`函数，`yield`关键字）

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



观察下面两种写法：

```
def encrypt_4(s: str, k: int = 3) -> str:  
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))  
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

```
sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))  
  
def encrypt_4(s: str, k: int = 3) -> str:  
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

练习：请分别给出解密代码

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



观察下面两种写法：

```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

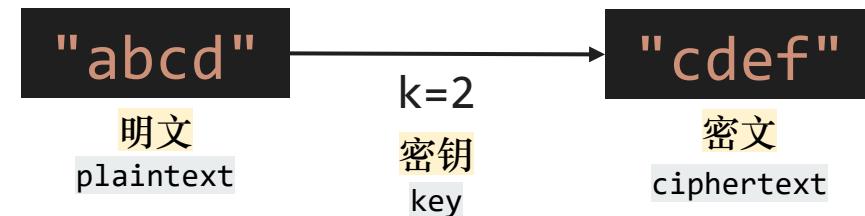
```
sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))

def encrypt_4(s: str, k: int = 3) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

上面两种写法的差异？

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



观察下面两种写法：

内置空间

```
def encrypt_4(s: str, k: int = 3) -> str:
    sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

局部空间

全局空间

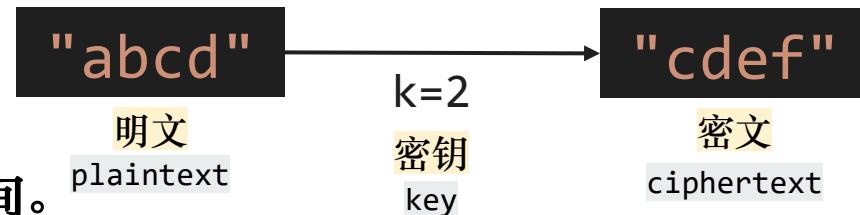
```
sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))

def encrypt_4(s: str, k: int = 3) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

上面两种写法的差异？

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



命名空间（Namespace）：我们所命名的事物单独享用其名字的空间。

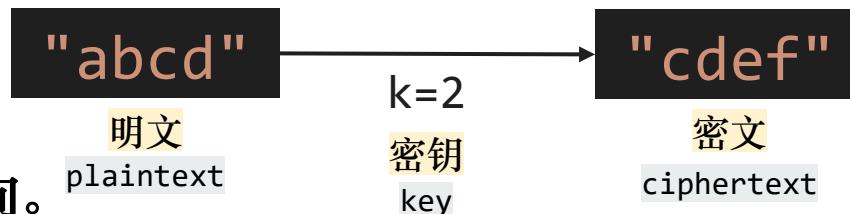


注：同一层缩进(`Tab`)（同一代码块）中的变量通常属于同一个局部命名空间，但并不是说同一层`Tab`就是同一个命名空间。

因为不同的代码块（即使缩进相同）可能属于不同的命名空间。

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



命名空间（Namespace）：我们所命名的事物单独享用其名字的空间。

作用域（Scope）：程序直接访问对应命名空间的代码区域。

内置作用域 Built-in Namespace

全局作用域 Global Namespace

内嵌作用域 Enclosing Namespace

局部作用域 Local Namespace

全局变量  
Global Variable

局部变量  
Local Variable

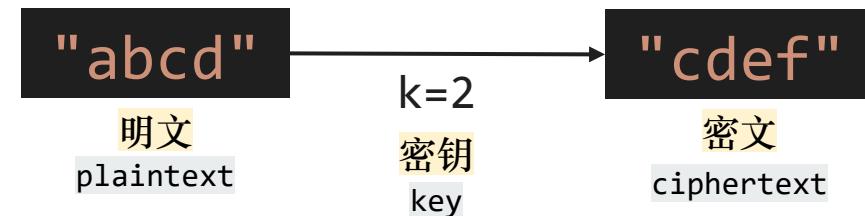
名可名，非常名。—老子《道德经》

```
a = 1
b = 2
for i in range(12):
    j = 3
    a = b + i
    print(a * i + j)
```

变量覆盖

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



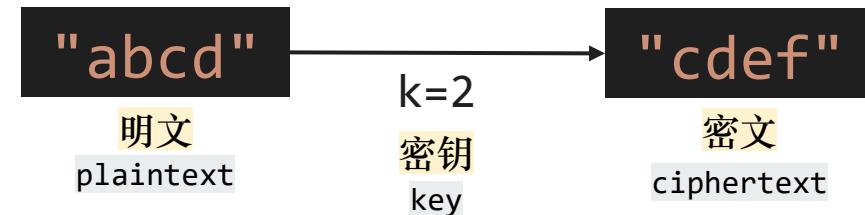
```
sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))

def encrypt(s: str, k: int = 3) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)
```

练习：请给出解密代码

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



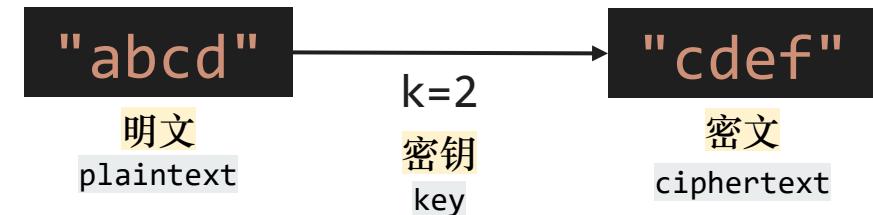
```
sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))

def encrypt(s: str, k: int = 3) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)

def decrypt(s: str, k: int = 3) -> str:
    return ''.join(sub(c, 'A', -k) if c.isupper() else sub(c, 'a', -k) if c.islower() else c for c in s)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



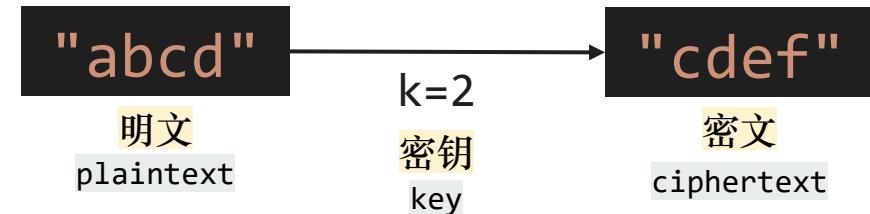
```
k: int = 3

sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
def encrypt(s: str) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)

def decrypt(s: str) -> str:
    return ''.join(sub(c, 'A', -k) if c.isupper() else sub(c, 'a', -k) if c.islower() else c for c in s)
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
k: int = 3

sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
def encrypt(s: str) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)

def decrypt(s: str) -> str:
    return ''.join(sub(c, 'A', -k) if c.isupper() else sub(c, 'a', -k) if c.islower() else c for c in s)
```

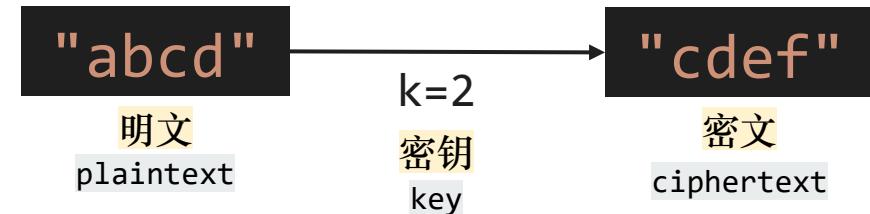
柯克霍夫原则 (Kerckhoffs' Principle)

密码方案的安全性不应该依赖于加密方案的保密性，而应当仅依赖于密钥的保密性。（保护容易，维护容易，检查容易）

Representer: Sike

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
k: int = 3

sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
def encrypt(s: str) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)

def decrypt(s: str) -> str:
    return ''.join(sub(c, 'A', -k) if c.isupper() else sub(c, 'a', -k) if c.islower() else c for c in s)
```

柯克霍夫原则 (Kerckhoffs' Principle)

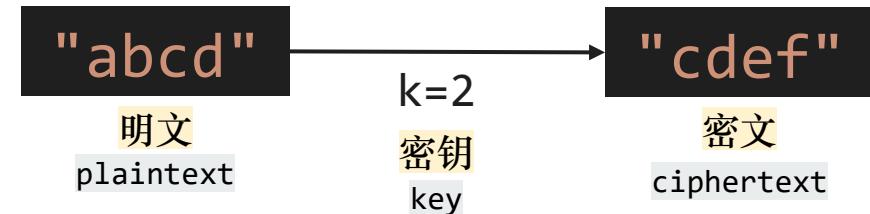
密码方案的安全性不应该依赖于加密方案  
的保密性，而应当仅依赖于密钥的保密性。 (保  
护容易，维护容易，检查容易)

→ 每一个用户都有自己的密钥

Representer: Sike

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



```
k: int = 3

sub = lambda ch, base, k: chr((ord(ch) - ord(base) + k) % 26 + ord(base))
def encrypt(s: str) -> str:
    return ''.join(sub(c, 'A', k) if c.isupper() else sub(c, 'a', k) if c.islower() else c for c in s)

def decrypt(s: str) -> str:
    return ''.join(sub(c, 'A', -k) if c.isupper() else sub(c, 'a', -k) if c.islower() else c for c in s)
```

柯克霍夫原则 (Kerckhoffs' Principle)

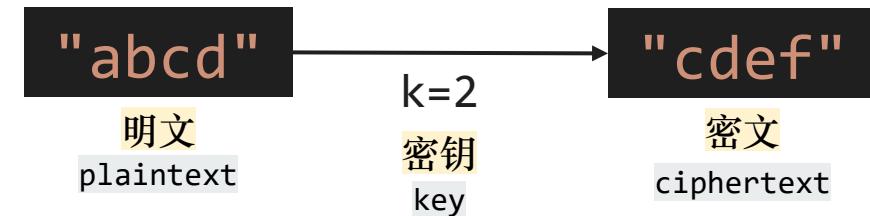
密码方案的安全性不应该依赖于加密方案的保密性，而应当仅依赖于密钥的保密性。（保护容易，维护容易，检查容易）

→ 每一个用户都有自己的密钥

在此处“框架”下对每个用户“分配”独有的密码方案。

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



在此处“框架”下对每个用户  
“分配”独有的密码方案。

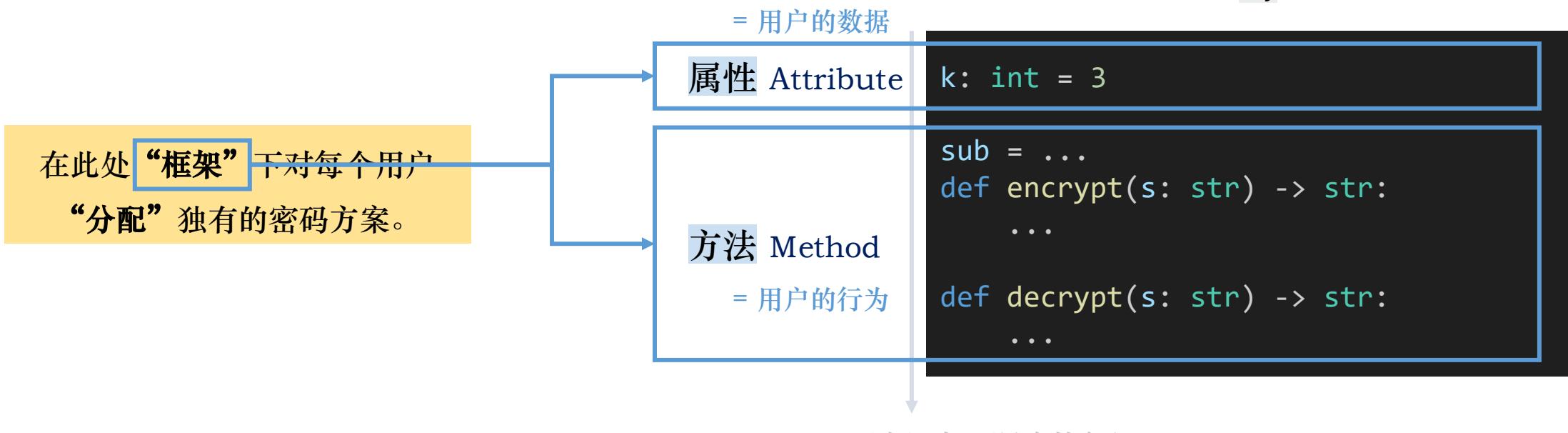
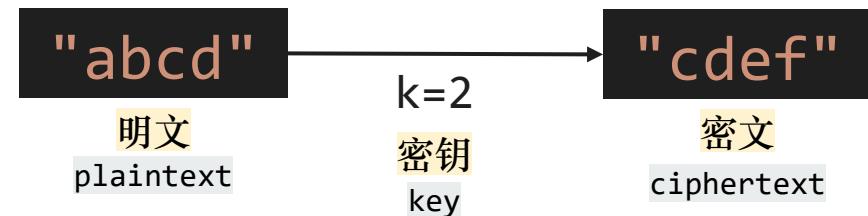
```
k: int = 3  
  
sub = ...  
def encrypt(s: str) -> str:  
    ...  
  
def decrypt(s: str) -> str:  
    ...
```

过程式（顺次执行）

全局变量 k 可能被修改  
函数之间的关系不清晰

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



全局变量 k 可能被修改  
函数之间的关系不清晰

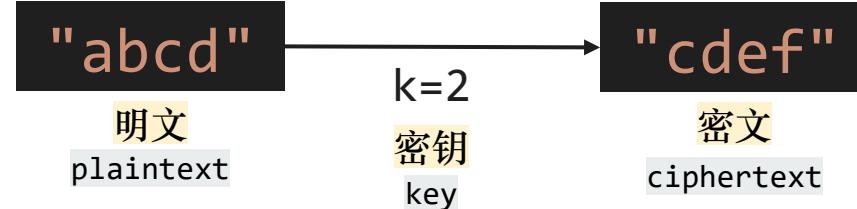
Representer: Sike

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

- 封装 (Encapsulation) : 将数据和方法包装在类中。

在此处“框架”下对每个用户  
“分配”独有的密码方案。

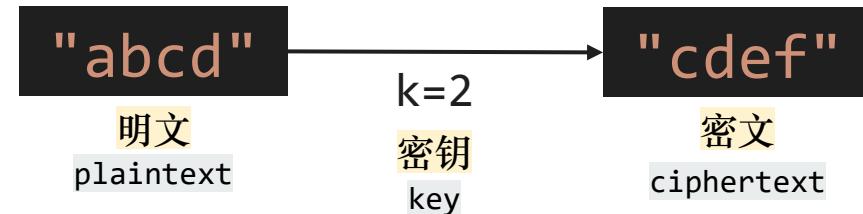
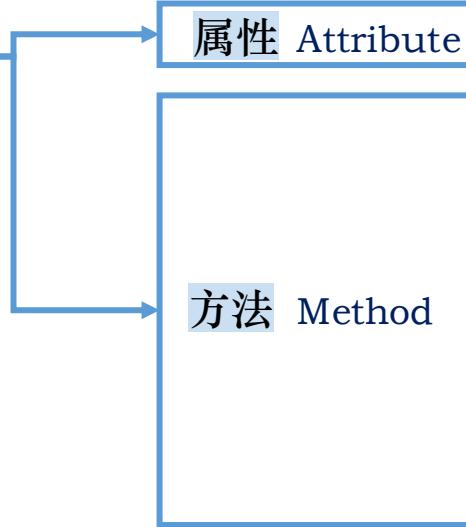


# Topic 2: Classical Crypto~

## 任务二：一个简单的加解密系统——以凯撒密码为例

- 封装 (Encapsulation) : 将数据和方法包装在类中;
- `self`: 指向自身的引用。

在此处“**框架**”下对每个用户  
“分配”独有的密码方案。



```

class Caesar: 构造函数 Constructor
    def __init__(self, k):
        self.k: int = k
        self.sub = ...

    def encrypt(self, s: str) -> str:
        ...

    def decrypt(self, s: str) -> str:
        ...
  
```

Code example for the Caesar class:

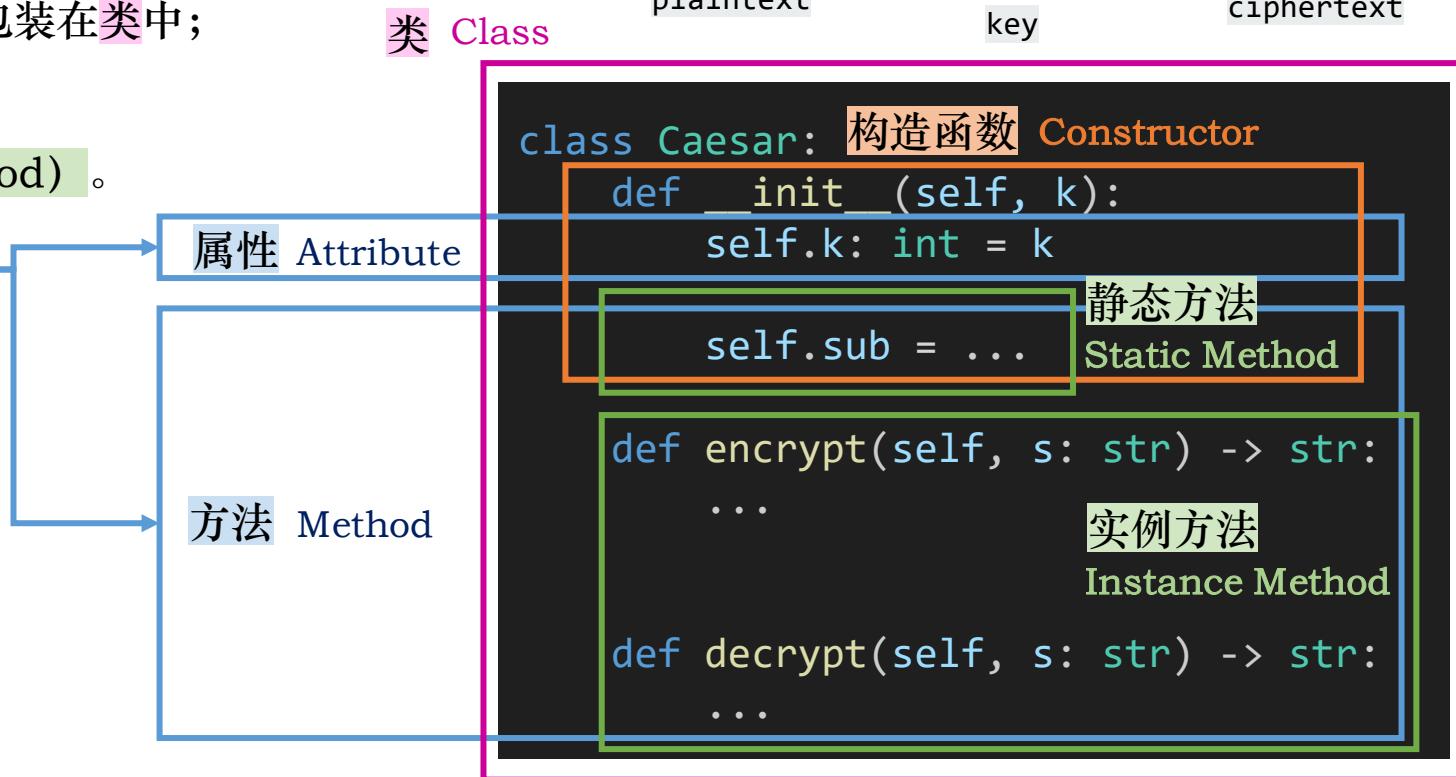
- `__init__` method: initializes the key (`k`) and creates a substitution table (`self.sub`).
- `encrypt` method: performs the encryption logic.
- `decrypt` method: performs the decryption logic.

# Topic 2: Classical Crypto~

## 任务二：一个简单的加解密系统——以凯撒密码为例

- 封装 (Encapsulation) : 将数据和方法包装在类中;
- `self`: 指向自身的引用;
- 静态方法/实例方法/类方法 (Class Method)。

在此处“框架”下对每个用户  
“分配”独有的密码方案。



# Topic 2: Classical Crypto~

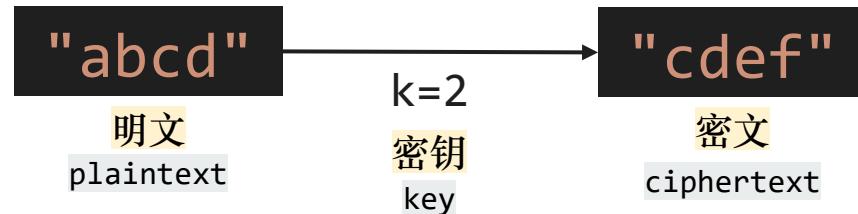
## 任务二：一个简单的加解密系统——以凯撒密码为例

- 封装 (Encapsulation)：将数据和方法包装在类中；
- `self`: 指向自身的引用；
- 静态方法/实例方法/类方法 (Class Method)。



- 对象：某一现实事物的软件（程序语言）下的抽象表达；
- 实例化：依据类创建对象的过程。

面向对象的编程 (Object-Oriented Programming)



类 Class

```
class Caesar:
    def __init__(self, k):
        ...
    def encrypt(self, s: str) -> str:
        ...
    def decrypt(self, s: str) -> str:
        ...
```

实例化

```
caesarUser1 = Caesar(7)
msgUser1 = "I am User One."
cfrUser1 = caesarUser1.encrypt(msgUser1)
de_cfrUser1 = caesarUser1.decrypt(cfrUser1)
```

方法调用

Represented: SKE

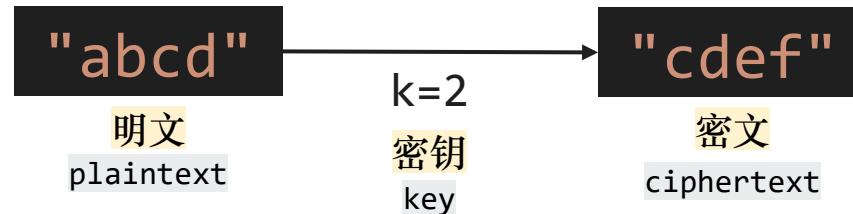
# Topic 2: Classical Crypto~

## 任务二：一个简单的加解密系统——以凯撒密码为例

- 封装 (Encapsulation)：将数据和方法包装在类中；
- `self`: 指向自身的引用；
- 静态方法/实例方法/类方法 (Class Method)。



- 对象：某一现实事物的软件（程序语言）下的抽象表达；
- 实例化：依据类创建对象的过程。



类 Class

```
class Caesar:
    def __init__(self, k):
        ...
    def encrypt(self, s: str) -> str:
        ...
    def decrypt(self, s: str) -> str:
        ...
```

实例化

```
caesarUser1 = Caesar(7)
msgUser1 = "I am User One."
cfrUser1 = caesarUser1.encrypt(msgUser1)
de_cfrUser1 = caesarUser1.decrypt(cfrUser1)
```

方法调用

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

我想要添加一些更实在的功能…

比如我想加密一个很长的文本文档…



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例

我想要添加一些更实在的功能…

比如我想加密一个很长的文本文档…

打开文件

操作文件

关闭文件

以只读方式打开

```
f = open(rpath, "r", encoding = "utf-8") → s = f.read()
```

返回str类型

```
f.close()
```

以只写方式打开

```
f = open(wpath, "w", encoding = "utf-8") → f.write(s)
```

输入str类型



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



打开文件

操作文件

关闭文件



! 为防止文件描述符/句柄的混乱，我们推荐使用 `with` 语句！

```
with open(rpath, "r", encoding = "utf-8") as f:
    m = f.read()
```

```
with open(wpath, "w", encoding = "utf-8") as f:
    f.write(c)
```

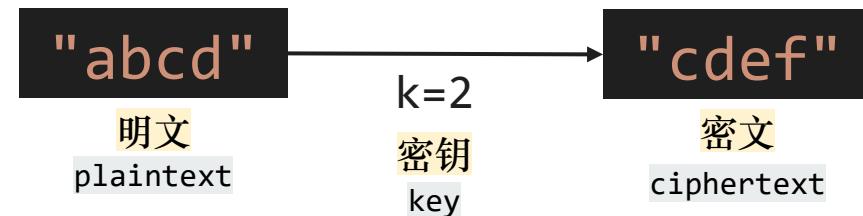
练习：完成下述两个函数，并添加到类中。

```
def encrypt_file(self, rpath, wpath):
    ... # 加密 rpath 的文件并写到 wpath
```

```
def decrypt_file(self, rpath, wpath):
    ... # 解密 rpath 的文件并写到 wpath
```

# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



- HTTP请求常用库
  - requests (pip install requests): 用于发送HTTP请求，简单易用。
  - urllib (built-in) : 用于发送HTTP请求，接口略复杂一些。
- Web应用构建常用库
  - flask (pip install flask): 创建 Web 服务器，提供 API 服务。
  - jinja2 (pip install jinja2): 生成动态HTML页面。
- 网页解析（爬虫）常用库
  - BeautifulSoup (pip install beautifulsoup4), scrapy, selenium ...

# Topic 2: Classical Crypto~

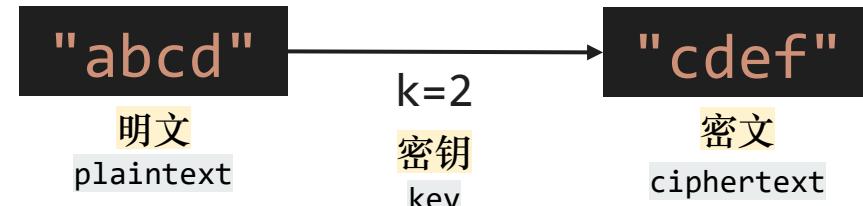
任务二：一个简单的加解密系统——以凯撒密码为例

tkinter, PyQt6, PySide6, ...



# Topic 2: Classical Crypto~

任务二：一个简单的加解密系统——以凯撒密码为例



- 拓展点三：修饰器，生成器
- 拓展点四：多文件项目
- 拓展点五：进程交互（pwntools）
- 拓展点六：AI工程（pytorch, sklearn...），Python绘图（matplotlib, seaborn...）...
- 应当熟悉的其他内容：一些底层的内容，如 t-string, f-string, list/tuple其他操作或性质（copy(), id(), 打包/解包），魔法函数...
- ...

# Review and Q&A

1. 匿名函数

**Anonymous Function/Lambda Expression**

2. 三元表达式

**Ternary Operator/Conditional Expression**

3. 海象运算符

**Walrus Operator/Assignment Expression**

4. 推导式

**Comprehension**

5. 命名空间与作用域

**Namespace & Scope**

6. 面向对象与类

**OOP & Class**

7. 文件读写

**File I/O (File Input/Output)**

8. 其他

**Miscellaneous**