BIOINFORMATICS · FOUNDATION

生信基础知识
从分子到数据

零基础入门生物信息学 · 为 Biopython 学习铺路

从细胞到分子 · 从序列到分析

适合完全零基础的生物信息学初学者

2026 年 8 月

目　录

前言

这门课讲什么

如果你想学会用 Python（以及 Biopython）分析生物数据，却对"DNA、基因、FASTA、BLAST"这些词感到陌生，那么这门课就是为你准备的。

本教程是一份零基础的生物信息学入门铺垫教材，它不要求你学过任何生物学知识，也不要求你写过一行代码。读完它，你将获得：

- 理解生物序列（DNA、RNA、蛋白质）到底"是什么"的能力；
- 看懂常见生物数据文件格式（FASTA、FASTQ、GenBank 等）的能力；
- 理解序列比对、数据库、BLAST、进化树等核心概念的能力；
- 一张"概念 → Biopython 工具"的对照地图，为接下来学习 Biopython 铺平道路。

> **提示** 本教程是《Biopython 零基础实战教程》的配套前置课程。建议按顺序学习：先读本教程掌握概念，再进入 Biopython 教程动手写代码。

适合谁

- **完全零基础**：没上过生物课、没写过代码，也没关系，我们从"细胞是什么"讲起；
- **会一点 Python 但不懂生物**：你可以直接跳到第 3 章以后，重点看序列与格式；
- **学生物但不会编程**：前三章你可以快速浏览，重点看第 4 章以后的"计算机视角"。

怎么学效果最好

1. **先概念，后工具**：不要一上来就背命令，先理解每个概念回答的是什么问题；
2. **边读边查**：遇到看不懂的英文缩写，翻到附录 B 的术语速查表；
3. **动手写**：第 3、5 章有少量 Python 演示，建议复制到自己的环境里运行一遍；
4. **做练习**：每章末尾有小练习，答案思路附在章末提示中，先自己想再做。

> **提示** 本教程中的数字（如人类基因组大小、氨基酸种类）采用主流教材的常用数值，具体细节可到第 7 章介绍的数据库中查阅原始数据。

第 1 章 生命的基本单元：从细胞到分子

1.1 细胞：生命的基本单位

地球上所有生命都由**细胞**组成。细胞大致分为两类：

- **原核细胞**：没有细胞核，遗传物质直接散在细胞质中，如细菌、古菌；
- **真核细胞**：有细胞核，遗传物质被核膜包裹，如动物、植物、真菌。

对生物信息学来说，细胞不是重点，**细胞里携带信息的分子**才是。每个细胞里都有三类与我们密切相关的分子：

分子	中文名	一句话角色
DNA	脱氧核糖核酸	遗传信息的"长期存储库"
RNA	核糖核酸	信息的"传递与执行者"
蛋白质	蛋白质	生命活动的"执行者"

生物信息学处理的"数据"，绝大多数就是从这三类分子中获得的**序列信息**——也就是用字母写下来的分子一级结构。

1.2 DNA：遗传物质

双螺旋结构

DNA（deoxyribonucleic acid，脱氧核糖核酸）是一种**双链螺旋**结构，1953 年由 Watson 与 Crick 提出。两条链反向平行缠绕，通过碱基配对连接。

四种碱基

DNA 由四种**碱基**组成，通常用四个字母表示：

- **A**：腺嘌呤（Adenine）
- **T**：胸腺嘧啶（Thymine）
- **G**：鸟嘌呤（Guanine）
- **C**：胞嘧啶（Cytosine）

碱基互补配对（关键规则）

两条链之间遵循严格的配对规则：

- **A 与 T 配对**（两个氢键）
- **G 与 C 配对**（三个氢键）

因此，如果一条链的序列是 `ATGC`，另一条与之互补的链就是 `TACG`。

> **注意** 这一规则是整个分子生物学的基石，也是程序里 `complement()`（互补）和 `reverse_complement()`（反向互补）函数的来源。学 Biopython 时你会反复用到它。

方向性：5' 到 3'

DNA 链是有方向的，一端叫 **5' 端**，另一端叫 **3' 端**。序列通常按 **5' → 3'** 方向书写和存储。

这带来一个重要的推论：DNA 双链反向平行，一条链的 5' 端对应另一条的 3' 端。计算反向互补序列（先互补、再反过来）就源于此。

1.3 基因与基因组

基因

基因是 DNA 上的一段具有功能意义的序列，通常编码蛋白质（或 RNA）。可以粗略地把基因理解为"一条指令"。

基因组

一个生物体的**全部遗传信息**（所有 DNA）叫**基因组**。几个参考数字：

生物	基因组大小（约）	蛋白编码基因数（约）
人类	31 亿碱基对（3.1 Gb）	2 万个
小鼠	27 亿碱基对	2.2 万个
大肠杆菌	460 万碱基对	4400 个
果蝇	1.4 亿碱基对	1.4 万个

> **注意** 基因组很大，但其中真正编码蛋白质的区域（外显子）只占人类基因组的约 1.5%。理解"序列很多、信息稀疏"这一点，对后续学习比对、注释很重要。

1.4 RNA：三种主要角色

RNA（核糖核酸）与 DNA 很像，但有两点不同：

- 糖不同（核糖 vs 脱氧核糖）；
- 碱基 **T 被 U（尿嘧啶）替代**，即 RNA 用 A、U、G、C 四个字母。

RNA 在细胞中有多种角色，主要记住三种：

类型	全称	作用
mRNA	信使 RNA	把 DNA 上的信息带到核糖体
tRNA	转运 RNA	搬运氨基酸，识别密码子
rRNA	核糖体 RNA	组成核糖体，催化翻译

生物信息学中最常处理的是 **mRNA / 转录本序列**——它是"DNA 与蛋白质之间的中间形态"。

1.5 蛋白质：分子机器

20 种氨基酸

蛋白质由 **20 种标准氨基酸**组成的长链。每种氨基酸用单字母或三字母缩写表示。例如：

单字母	三字母	中文名	单字母	三字母	中文名
A	Ala	丙氨酸	M	Met	甲硫氨酸
R	Arg	精氨酸	F	Phe	苯丙氨酸
N	Asn	天冬酰胺	P	Pro	脯氨酸
D	Asp	天冬氨酸	S	Ser	丝氨酸
C	Cys	半胱氨酸	T	Thr	苏氨酸
E	Glu	谷氨酸	W	Trp	色氨酸
Q	Gln	谷氨酰胺	Y	Tyr	酪氨酸
G	Gly	甘氨酸	V	Val	缬氨酸
H	His	组氨酸	K	Lys	赖氨酸
I	Ile	异亮氨酸	L	Leu	亮氨酸

> **提示** 你不必现在背下 20 种氨基酸，但至少要认识 `A R N D C Q E G H I L K M F P S T W Y V` 这 20 个字母，因为**蛋白序列文件里就是这些字母**。

蛋白质的结构层次

- **一级结构**：氨基酸的排列顺序（就是"序列"，生物信息学主要处理这一层）；
- **二级结构**：局部的 α 螺旋、β 折叠等规则构象；
- **三级结构**：整条链的空间折叠（PDB 数据库存储的就是这一层）；
- **四级结构**：多条链组装成复合物。

序列 → 结构 → 功能

蛋白质的功能由它的三维结构决定，而结构由氨基酸序列决定（大致如此）。因此**序列信息是理解生命的最基本输入**——这也是为什么生物信息学从"序列"开始。

1.6 小结与练习

小结

- 细胞分原核与真核；DNA、RNA、蛋白质是三大关键分子；
- DNA 用 A/T/G/C 四个字母，遵守 A-T、G-C 配对，有 5'→3' 方向；
- 基因是一段有功能的 DNA，基因组是全部遗传信息；
- RNA 把 T 换成 U，mRNA 是中间信使；
- 蛋白质由 20 种氨基酸组成，一级结构就是序列。

练习

1. 写出 `ATGC` 的互补序列和反向互补序列（提示：先互补，再倒序）。
2. 一段双链 DNA 中 A 占 30%，那么 G 大约占多少？（提示：A=T，G=C）
3. RNA 与 DNA 的字母表差在哪里？

第 2 章 中心法则：DNA → RNA → 蛋白质

2.1 中心法则总览

中心法则（Central Dogma）描述遗传信息的流动方向：

```
DNA --(转录)--> RNA --(翻译)--> 蛋白质
```

- **复制**（Replication）：DNA → DNA；
- **转录**（Transcription）：DNA → mRNA；
- **翻译**（Translation）：mRNA → 蛋白质。

这条法则解释了"信息如何从存储形式变成执行形式"，也是序列分析中"转录、翻译"这两个操作（Biopython 中 `Seq.transcribe()` 和 `Seq.translate()`）的生物学依据。

2.2 DNA 复制

DNA 复制时，双链解开，各自作为模板合成互补新链，最终一个 DNA 分子变成两个完全相同的分子。

对生物信息学来说，复制带来的启示是：**序列的"反向互补"是真实存在的生物学操作**，而不是程序员的怪癖。

2.3 转录：DNA → mRNA

转录是"读模板、写 mRNA"的过程：

- 以 DNA 的一条链为**模板链**（template strand），合成 mRNA；
- mRNA 与模板链互补，因此与另一条链（**编码链**）序列一致（只是 T 换成了 U）；
- 一个基因转录出的 mRNA 长度通常与基因的编码区相当。

举例：假设编码链（5'→3'）是

```
ATGGCCATTGTA
```

对应 mRNA（5'→3'）为

```
AUGGCCAUUGUA
```

> **提示** 这就是 Biopython 中 `Seq.transcribe()` 做的事情：把 DNA 序列里的 `T` 替换成 `U`。听起来简单，但它是中心法则在代码里的体现。

2.4 翻译：mRNA → 蛋白质

翻译是"读 mRNA、拼蛋白质"的过程：

1. 核糖体从 mRNA 的 5' 端开始，**每 3 个碱基一组**（称为一个**密码子**）；
2. 每个密码子对应一种氨基酸（或终止信号）；
3. 从起始密码子 **AUG** 开始，到终止密码子 **UAA / UAG / UGA** 结束。

把上一节的 mRNA 翻译一遍：

```
AUG GCC AUU GUA
 ↓   ↓   ↓   ↓
Met  Ala  Ile  Val
```

蛋白质序列写作：`MAIV`（用单字母缩写）。

读框（Reading Frame）

因为密码子每次读 3 个碱基，从不同位置开始读，会得到完全不同的结果：

- 读框 1：从第 1 个碱基开始；
- 读框 2：从第 2 个碱基开始；
- 读框 3：从第 3 个碱基开始。

一条 DNA 有 3 个正向读框、3 个反向读框，共 6 个读框。**寻找编码区域（ORF）**就是在这 6 个读框中找"从起始密码子到终止密码子、能编码较长蛋白"的片段——这是基因预测的基础。

2.5 遗传密码表（简版）

遗传密码表共有 64 个密码子：61 个编码氨基酸，3 个是终止密码子。下面是常用的几个：

密码子	氨基酸	说明
AUG	M（甲硫氨酸）	起始密码子
UAA / UAG / UGA	（终止）	终止密码子
UUU / UUC	F（苯丙氨酸）	—
GGU / GGC / GGA / GGG	G（甘氨酸）	—
UCU / UCC / UCA / UCG	S（丝氨酸）	—
GCU / GCC / GCA / GCG	A（丙氨酸）	—

> **提示** 完整 64 密码子表见附录 A。学 Biopython 时 `Seq.translate()` 内部就是这张表，你也可以自定义表格。

密码子的简并性

61 个密码子只编码 20 种氨基酸，说明**多种密码子可以对应同一种氨基酸**，这称为密码子的**简并性**（degeneracy）。例如亮氨酸 L 有 6 个密码子。简并性让突变常常"沉默"——碱基变了，氨基酸却没变。

2.6 突变与序列变异

突变指 DNA 序列发生变化，常见类型：

类型	说明	例子
点突变（替换）	一个碱基被另一个替换	A → G
插入（Insertion）	插入一段碱基	中间多了 CG
缺失（Deletion）	删掉一段碱基	少了 A
倒位（Inversion）	一段序列方向颠倒	ATGC → CGTA

突变对蛋白质的影响

- **同义突变**：碱基变了，氨基酸不变（利用简并性）；
- **错义突变**：氨基酸改变（可能影响功能）；
- **无义突变**：提前变成终止密码子，蛋白截短（通常有害）；
- **移码突变**：插入/缺失使读框错位（通常影响巨大）。

变异（Variation）是群体层面的说法：不同个体的序列差异，例如 SNP（单核苷酸多态性）。序列比对正是发现这些差异的利器。

2.7 小结与练习

小结

- 中心法则：DNA 复制 → 转录成 RNA → 翻译成蛋白质；
- 转录只是 T→U；翻译每 3 个碱基一个密码子；
- AUG 起始，UAA/UAG/UGA 终止；一条序列有 6 个读框；

· 密码子有简并性，突变分同义/错义/无义/移码。

练习

1. 把 DNA `ATGGCCATTGTA` 先转录再翻译，写出氨基酸序列（用单字母）。
2. 为什么插入一个碱基常常"破坏"整个蛋白？
3. 如果一段 mRNA 以 `UGA` 开头，它会编码蛋白吗？

第 3 章 序列数据：计算机眼中的生物学

3.1 为什么序列可以用字符串表示

在上一章我们看到：DNA 是 A/T/G/C 的线性排列，蛋白质是 20 种氨基酸的线性排列。这带来一个关键洞察——

> **生物序列本质上就是字符串。** 对计算机而言，一条 DNA 就是一个只含 `A`、`T`、`G`、`C` 的字符串；一条蛋白质就是一个含 20 种字母的字符串。

这正是生物信息学能与编程结合的根基：**字符串处理能力 = 序列分析能力**。你学过的 Python 字符串方法（索引、切片、拼接、查找），几乎都能直接用在序列上。

3.2 核酸字母表（IUPAC 代码）

核酸序列不一定永远只有 A/T/G/C。测序结果中经常出现模糊碱基，IUPAC 规定了扩充字母表：

字母	含义	代表哪些碱基
A	腺嘌呤	A
T	胸腺嘧啶	T
G	鸟嘌呤	G
C	胞嘧啶	C
U	尿嘧啶（RNA）	U
R	嘌呤	A 或 G
Y	嘧啶	C 或 T
N	任意碱基	A/T/G/C 之一
S	强键（三氢键）	G 或 C
W	弱键（两氢键）	A 或 T
K	酮基	G 或 T
M	氨基	A 或 C
B	非 A	C/G/T
D	非 C	A/G/T
H	非 G	A/C/T
V	非 T	A/C/G

> **提示** 处理测序数据时最常见的是 **N**（该位置无法确定）。很多分析脚本的第一步就是统计序列里有多少个 N。

3.3 蛋白质字母表

蛋白质序列使用 **20 个标准氨基酸字母**，外加两个特殊符号：

符号	含义
B	Asx：天冬氨酸或天冬酰胺（不确定时用）
Z	Glx：谷氨酸或谷氨酰胺（不确定时用）
X	任意氨基酸
*	终止符号（翻译结果中表示终止密码子）

> **注意** 在 Biopython 里，`Seq.translate()` 的结果中常出现 `*`，表示蛋白序列的终止位置。ORF 查找本质上就是"用 `*` 切分翻译结果"。

3.4 方向约定：序列按 5' → 3' 存储

几乎所有公共数据库的核酸序列都按 5' → 3' 方向存储。这意味着：

- FASTA 文件里的 `ATGC` 表示 5'-ATGC-3'；
- 想得到与之互补的 DNA 链，需要"先互补再反转"（即反向互补）。

这也是为什么 Biopython 会同时提供 `complement()` 和 `reverse_complement()` 两个方法——后者才是真正代表"另一条链"的操作。

3.5 用 Python 感受一下（必读示例）

下面的代码**不依赖 Biopython**，只用 Python 字符串就能完成序列的基本操作。请在你的环境里运行一遍：

```python
# 序列就是字符串
dna = "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG"

print("长度:", len(dna))                    # 序列长度
print("前 10 个碱基:", dna[:10])             # 切片
print("A 的数量:", dna.count("A"))           # 计数
print("GC 数量:", dna.count("G") + dna.count("C"))

# 互补：A<->T，G<->C
complement_map = str.maketrans("ATGC", "TACG")
complement = dna.translate(complement_map)
print("互补序列:", complement)

# 反向互补（模拟另一条链）
reverse_complement = complement[::-1]
print("反向互补:", reverse_complement)

# 转录（DNA -> mRNA，把 T 换成 U）
mrna = dna.replace("T", "U")
print("mRNA:", mrna)
```

输出

```
长度：39
前 10 个碱基：ATGGCCATTG
A 的数量：9
GC 数量：22
互补序列：TACCGGTAACATTACCCGGCGACTTTCCCACGGGCTATC
反向互补：CTATCGGGCACCCTTTCAGCGGCCCATTACAATGGCCAT
mRNA：AUGGCCAUUGUAAUGGGCCGCUGAAAGGGUGCCCGAUAG
```

> **提示** 这段代码里"先互补、再反转"的顺序，正是 Biopython `Seq.reverse_complement()` 内部做的事。理解它，你就理解了序列操作最核心的一步。

练习建议

把上面的代码改成函数形式：

```python
def reverse_complement(seq):
    return seq.translate(str.maketrans("ATGC", "TACG"))[::-1]
```

3.6 小结与练习

小结

- 序列 = 字符串，字符串操作 = 序列操作；
- 核酸字母表含模糊碱基（N 最常见）；
- 蛋白字母表含 X、* 等符号；
- 公共数据按 5'→3' 存储；反向互补 = 先互补再反转。

练习

1. 手写函数 `gc_content(seq)`，返回 GC 占序列的百分比。
2. `"ATNNCG"` 中 N 的占比是多少？这种序列能否直接翻译？
3. 为什么数据库中几乎不用"3'→5' 方向"存储序列？

第 4 章 生物序列文件格式：你与数据打交道的起点

4.1 为什么必须先懂格式

生物数据以**文件**形式存放，而文件有固定格式。**不懂格式，就没法正确读取数据；格式理解错了，分析结果就是错的**。

对后续学习特别重要的一点是：Biopython 的 `SeqIO` 模块就是"格式转换器"——它负责把各种格式的文件读成统一的对象。你越了解格式，用起 `SeqIO` 越顺手。

> **核心思想** 所有序列格式都在回答四个问题：**这条序列叫什么？序列是什么？有没有额外信息（注释/质量）？多序列如何组织？**

4.2 FASTA：最基础的序列格式

FASTA 是**最简单、最通用**的序列格式，几乎任何工具都支持。

```
>seq1 人类胰岛素基因部分序列
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG
```

规则：

- 以 `>` 开头的行是**标题行**：`>` 后紧跟序列标识符（id），空格后的内容通常叫描述（description）；
- 标题行之后是**序列行**，可以连续多行；
- 核酸或蛋白均可存放。

在 Biopython 中，FASTA 的标题行对应 `SeqRecord.id` 和 `SeqRecord.description`，序列行对应 `SeqRecord.seq`。

> **注意** FASTA 没有质量信息、没有注释块，因此信息量最小。它是"存序列"而不是"描述序列"的格式。

4.3 FASTQ：带质量的测序数据格式

高通量测序仪输出的**原始数据**几乎都是 FASTQ。每条记录固定 4 行：

```
@SRR001666.1 071112_SLXA-EAS1_s_7:5:1:817:345 length=36
GGGGATGTCGTTGTGCTGCTATTGCTGGCCGGTCGCAG
```

```
+
IIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIIII
```

四行含义：

行	前缀	内容
1	@	序列标识符 + 注释
2	—	碱基序列
3	+	通常只有 +（可重复标识符）
4	—	质量值字符串（与序列等长）

质量值（Phred 得分）

第 4 行的每个字符都代表该位置碱基的质量：

- ASCII 码转换为 Phred 得分：`Q = ASCII码 - 33`（常见 Sanger 编码）；
- Q 与错误概率的关系：`Q = -10 × log10(P错误)`；
- Q=30 表示错误概率千分之一，是常用的质量阈值。

> **提示** 质量值是"二选一"的典型例子：`Q20`（错误率 1%）和 `Q30`（错误率 0.1%）是测序质控（如 FastQC）里最常见的两个指标。Biopython 的 `SeqIO` 可以把 FASTQ 读成带 `letter_annotations["phred_quality"]` 的 `SeqRecord`。

4.4 GenBank：带注释的序列格式

GenBank 格式（GenBank/GenPept 格式）是**信息最丰富**的核酸序列格式之一。文件从 LOCUS 行开始，包含大量注释：

输出

```
LOCUS       NM_000207       5112 bp    mRNA    linear   PRI 05-NOV-2021
DEFINITION  Homo sapiens insulin (INS), mRNA.
ACCESSION   NM_000207
VERSION     NM_000207.3
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
FEATURES             Location/Qualifiers
     gene            1..5112
                     /gene="INS"
     CDS             join(1..201,561..1019)
                     /codon_start=1
                     /product="insulin preproprotein"
                     /protein_id="NP_000198.1"
ORIGIN
        1 atggccctgt ggatgcgcct cctgcccctg ctggcgctgc tggccctctg
       61 gggacctgac ccagccgcag cctttgtgaa ccaacacctg tgcggctcac
//
```

关键字段：

字段	含义
LOCUS	名称、长度、类型（mRNA/DNA...）、拓扑
DEFINITION	一句话描述
ACCESSION / VERSION	稳定的标识符（如 NM_000207.3）
FEATURES	注释区：gene、CDS、mRNA 等特征及其位置
ORIGIN	序列正文（按 60 个碱基一行分行显示）
//	记录结束符

> **注意** GenBank 里的位置表示法 `join(1..201,561..1019)` 表示"CDS 由两段拼接而成"——这在真核生物中很常见（内含子被剪接掉）。Biopython 的 `SeqFeature` 就负责解析这类复杂位置。

4.5 EMBL：欧洲风格的注释格式

EMBL 格式（欧洲生物信息学研究所 EBI 采用）与 GenBank 类似，但字段风格不同，例如用 `ID`、`FT`（feature）等两字母行前缀，序列行以 `SQ` 开始、以 `//` 结束：

输出

```
ID   INS_HUMAN     standard; RNA; PRI; 5112 BP.
AC   X70507;
DE   Homo sapiens mRNA for insulin.
FT   CDS             join(1..201,561..1019)
FT                   /gene="INS"
SQ   Sequence 5112 BP;
     atggccctgt ggatgcgcct cctgcccctg ctggcgctgc tggccctctg
//
```

Biopython 的 `SeqIO.parse(handle, "embl")` 可以直接读取，读取后的对象与 GenBank 完全相同。

4.6 比对格式：Clustal / Stockholm

多序列比对结果也有专门格式。最常见的是 **Clustal 格式**：

输出

```
CLUSTAL W multiple sequence alignment


seqA  ACGTACGTACGTACGT
seqB  ACGTACGTACGTACGA
seqC  ACGTTCGTACGTACGT
      ** **  ** ******
```

- 第一行是格式标识；
- 每行：序列名 + 一段序列；
- 最后一行 `*` 表示该列保守（完全一致），`:` 表示相似，`.` 表示一般相似。

> **提示** 在 Biopython 里，多序列比对文件用 `AlignIO` 读写（`AlignIO.read/parse`），与 `SeqIO` 处理单序列是姊妹关系。**格式决定用哪个模块**：序列 → SeqIO；比对 → AlignIO。

4.7 格式转换：思路比工具更重要

几乎所有分析的第一步都是"把数据读成统一格式"。常见的转换场景：

从	到	场景
FASTQ	FASTA	去掉质量值，只保留序列
GenBank	FASTA	只提取序列用于比对
FASTA	GenBank	为序列添加注释
Clustal	Stockholm	换一个比对查看器

在 Biopython 中，这类转换通常只需要两行代码（`SeqIO.parse` + `SeqIO.write`），但前提是你**认识这些格式**。这正是本章的目的。

4.8 小结与练习

小结

- 四种必知格式：FASTA（纯序列）、FASTQ（序列+质量）、GenBank/EMBL（序列+注释）、Clustal（比对）；
- 格式决定读取方式：`SeqIO` 管序列，`AlignIO` 管比对；
- FASTQ 的质量值是 Phred 得分（ASCII 减 33）；
- GenBank 的 FEATURES 提供基因、CDS 等注释，`//` 是记录结束符。

练习

1. 指出下面是什么格式，并说出它的四行结构：`@x` / `ACGT` / `+` / `IIII`。
2. FASTA 标题行 `>g1 human gene` 中，`id` 和 `description` 分别是什么？
3. GenBank 中 `join(1..201,561..1019)` 表达什么生物学含义？

第 5 章 从序列到知识：统计与特征

5.1 GC 含量：序列最基本的统计量

GC 含量 = (G + C 的数量) ÷ 序列总长度。它是序列分析中最基础、最常用的指标之一。

为什么关注 GC?

- **稳定性**：G-C 之间有三个氢键，A-T 只有两个。GC 含量高的区域 DNA 更耐热；
- **物种特征**：不同物种基因组 GC 含量差异明显（人类约 41%，大肠杆菌约 51%，疟原虫约 19%）；
- **实验意义**：PCR 引物设计、测序覆盖度都与 GC 相关；
- **编码区信号**：蛋白编码区通常 GC 含量高于非编码区（可作为基因预测的线索之一）。

用 Python 计算（手动版）

```python
seq = "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG"
gc = (seq.count("G") + seq.count("C")) / len(seq)
print(f"GC 含量: {gc * 100:.1f}%")
```

输出

```
GC 含量: 56.4%
```

> **提示** 在 Biopython 中，`Bio.SeqUtils.gc_fraction()` 返回 0~1 之间的小数；旧版 `gc()` 返回百分数。学 Biopython 时注意区分版本差异。

5.2 序列长度与组成

统计一个数据集的基本情况，是任何分析的第一步：

指标	含义	常见用途
序列长度	碱基/氨基酸个数	判断数据类型（如读长 150 bp）
碱基频率	A/T/G/C 各自占比	初步质控
N 数量	无法确定的位点	序列质量
序列条数	文件里有多少条序列	数据集规模

> **注意** 一条测序读长通常 50~250 bp，而一条完整基因可达数千 bp。**长度差异常常提示数据类型**（原始测序 reads vs 组装好的 contigs vs 完整基因）。

5.3 子序列与模体

模体（Motif）是序列中反复出现、具有生物学意义的小片段，例如：

- 启动子区域的 `TATA` 盒（TATAAT）；
- 蛋白质里的**结构域**基序，如锌指 `C2H2`；
- 酶切位点，如限制酶 EcoRI 识别 `GAATTC`。

在序列中查找模体，本质就是**字符串查找**：

Python

```python
seq = "AATTCGAATTCGGGGAATTCC"
motif = "GAATTC"
pos = []
start = 0
while True:
    idx = seq.find(motif, start)
    if idx == -1:
        break
    pos.append(idx + 1)  # 生物学中常用 1 起始坐标
    start = idx + 1
print("模体出现位置:", pos)
```

输出

```
模体出现位置: [6, 15]
```

> **提示** 这个"找模体"的逻辑，对应 Biopython 的 `Bio.motifs` 模块和 `SeqUtils.nt_search()`。理解字符串查找，就理解了模体分析的一半。

5.4 密码子、读框与 ORF

在第 2 章我们知道：每 3 个碱基一个密码子，从不同起点开始产生不同的读框。**ORF（Open Reading Frame，开放阅读框）**指一条从起始密码子开始、到终止密码子结束、没有中途终止的连续编码片段。

为什么要找 ORF?

- 预测新基因（在基因组中找候选编码区）；
- 验证注释（已有基因应该对应一个长 ORF）；
- 分析宏基因组/转录组数据。

ORF 查找的直觉（Python 演示）

```python
def translate6(dna):
    """把 DNA 的 6 个读框都翻译出来"""
    table = str.maketrans("ATCG", "TAGC")
    strands = [dna, dna.translate(table)[::-1]]  # 正向 + 反向互补
    for i, strand in enumerate(strands):
        for frame in range(3):
            codons = [strand[j:j+3] for j in range(frame, len(strand)-2, 3)]
            yield f"链{'+' if i==0 else '-'} 读框{frame+1}", codons

dna = "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG"
for label, codons in translate6(dna):
    print(label, "→", " ".join(codons[:6]), "...")
```

输出

```
链+ 读框1 → ATG GCC ATT GTA ATG GGC CGC ...
链+ 读框2 → TGG CCA TTG TAA TGG GCC GCT ...
链+ 读框3 → GGC CAT TGT AAT GGG CCG CTG ...
链- 读框1 → CTA TCG GGC ACC CTT TCA GCC GCC ...
链- 读框2 → TAT CGG GCA CCC TTT CAG CCG CCC ...
链- 读框3 → ATC GGG CAC CCT TTC AGC CGC CCA ...
```

注意到读框 2 中出现了 `TAA`——这是终止密码子，说明这个读框不编码。**找 ORF 就是找"从 AUG 到终止密码子、中间无终止"的片段**，Biopython 的 `Seq.translate()` 会让这个过程变得非常简单。

5.5 蛋白质理化性质

分析蛋白序列时，常用理化性质包括：

性质	含义	用途
分子量	氨基酸分子量之和（近似）	鉴定蛋白、质谱对照
等电点 pI	蛋白净电荷为零时的 pH	纯化、电泳条件设计
疏水性（GRAVY）	平均亲水/疏水指数	预测跨膜区域
氨基酸组成	各氨基酸比例	功能线索
不稳定指数	蛋白在体内的稳定性预测	实验参考

> **提示** Biopython 的 `Bio.SeqUtils.ProtParam.ProteinAnalysis` 可以一行算出所有这些指标。但在用工具之前，理解"分子量是氨基酸质量之和"这个直觉更重要。

5.6 小结与练习

小结

- GC 含量是最基础的序列统计量，有实验和进化意义；
- 模体查找 = 字符串查找；
- ORF 是从起始到终止、无中途终止的编码片段，有 6 个读框；
- 蛋白性质（分子量、pI、疏水性）是序列分析的重要产出。

练习

1. 计算 `GGCC` 的 GC 含量。
2. 一段 DNA 在第 2 读框出现 `TGA`，说明什么？
3. 为什么 GRAVY 值高的蛋白更可能跨越细胞膜？

第 6 章 序列比对：寻找相似与同源

6.1 为什么要比对

比对（Alignment）就是把两条或多条序列"排排站"，让相似的位点对齐，从而：

- 找出序列之间的相似区域；
- 推断它们是否来自共同祖先（同源）；
- 发现保守的功能位点；
- 为新序列的功能注释提供依据。

看一个直观的例子：

```
序列 A:  A C G T A C G T
序列 B:  A C G T A C G A
                  (不匹配)
```

这两条序列高度相似，只有最后一个位置不同。比对让这个差异一目了然。

6.2 相似性 ≠ 同源性（关键概念）

这是初学者最容易混淆的一对概念：

概念	含义	性质
相似性（Similarity）	序列在数值上有多像（可量化）	可测量、可比较
同源性（Homology）	是否来自共同祖先	二元的：同源或不同源，没有程度之分

> **注意** 说"两条序列 80% 同源"是**错误的**——同源没有百分比。应该说"80% 相似"，然后推断"可能同源"。许多文献误用这两个词，请务必区分。

同源关系细分

- **直系同源（Ortholog）**：不同物种中由共同祖先基因分化而来，功能通常相似；
- **旁系同源（Paralog）**：同一物种内由基因复制产生，功能可能分化。

6.3 双序列比对：全局 vs 局部

全局比对（Global）：从开头对齐到结尾，适合长度相近的序列。

```
ACGTACGT
AC--ACGA   （引入 gap 补偿长度差异）
```

局部比对（Local）：只找最相似的局部片段，适合"在一个长序列里找一小段相似区"。

在 Biopython 中对应 `PairwiseAligner` 的 `mode` 参数：`"global"` 和 `"local"`。

Gap（缺口）

比对中常出现 `-`（gap），表示"这条序列在这个位置缺失了碱基"。gap 的产生原因包括插入/缺失突变（indel）。**gap 通常需要惩罚**——比对算法在"允许错配"和"开 gap"之间权衡，这就是打分。

6.4 打分矩阵：氨基酸之间怎么算分

DNA 比对常用简单规则：匹配 +1、错配 0（或 -1）。但**蛋白质比对更复杂**：不同氨基酸替换成另一个氨基酸的"代价"不同（化学性质相近的替换更常见）。

替换矩阵（Substitution Matrix）给出任意两个氨基酸的得分。两个经典系列：

矩阵	特点	适用场景
BLOSUM62	基于蛋白质家族的保守区块	一般用途（默认选择）
BLOSUM50 / 80	保守程度不同	亲缘远 / 近
PAM250	基于进化模型	早期经典，远缘序列

BLOSUM62 的部分值（分数越高越易替换）：

	A	S	T	K
A	4	1	0	-1
S	1	4	1	0
T	0	1	5	0
K	-1	0	0	5

> **提示** 观察对角线：同氨基酸得分最高；带相似侧链的（如 S/T）得分高于不相似的。理解"得分高 = 更容易替换"就够用了，矩阵的具体数值由工具处理。

6.5 多序列比对（MSA）

把**多条**序列同时对齐叫多序列比对，例如 ClustalW 输出：

```
seqA   ACGTACGTACGT
seqB   ACGTACGTACGA
seqC   ACGTTCGTACGT
       ** **  *****
```

`*` **表示保守列（所有序列一致），** `:` **表示相似列**。保守列往往对应功能重要的位点。

多序列比对的意义：

- 找出进化上保守的区域（功能位点）；
- 作为构建进化树的输入；
- 训练蛋白质结构/功能预测模型。

6.6 怎么读懂一次比对结果

当你拿到 BLAST 或比对工具的输出时，重点看四个指标：

指标	含义
Identity（一致性）	完全匹配的位点比例
Similarity（相似性）	匹配+保守替换的比例
Coverage（覆盖度）	比对上区域的长度占序列总长的比例
Score（得分）	综合打分（含 gap 惩罚）

高覆盖 + 高一致 ≈ 强烈同源信号；反之，短片段的高一致可能是偶然。

6.7 小结与练习

小结

- 比对是把序列对齐、找相似区域的工具；
- 相似性是数值，同源性是性质，二者不可混用；
- 全局 vs 局部、gap 惩罚、替换矩阵是比对的核心要素；
- 多序列比对用 `*` / `:` 标记保守列；
- 读结果看 identity、coverage、score。

练习

1. 判断题：两序列 90% 同源。（对/错，为什么？）
2. 全局比对和局部比对各自适合什么场景?
3. BLOSUM62 对角线上为什么分数最高?

第 7 章 生物数据库：数据从哪里来

7.1 三大国际数据中心

全世界的生物序列数据主要存放在三大机构，它们之间每天同步数据：

机构	全称	侧重
NCBI（美国）	美国国家生物技术信息中心	GenBank、PubMed 等
EBI（欧洲）	欧洲生物信息学研究所	EMBL、UniProt 等
DDBJ（日本）	日本 DNA 数据库	亚洲提交的序列

> **提示** 三大中心的数据互为镜像，检索任意一个即可。NCBI 在生物信息学教学中最常用，Biopython 的 `Entrez` 模块就是对接 NCBI 的。

7.2 核酸数据库：GenBank 与 RefSeq

GenBank

- 收录**研究者直接提交**的原始序列（可能含冗余）；
- 记录用 accession（如 `AB000001`）标识；
- 适合：查找特定提交的序列。

RefSeq

- NCBI 的**人工整理/自动精选**参考序列库；
- 标识符带前缀，如 `NM_`（mRNA）、`NP_`（蛋白）、`NC_`（染色体）、`NG_`（基因区）；
- 冗余低、注释质量高，**日常分析首选**。

示例：人胰岛素基因 mRNA 的 RefSeq 编号是 **NM_000207.3**。

> **注意** `NM_` 是 mRNA，`NP_` 是对应蛋白。看到标识符前缀，就能猜出分子类型——这是很实用的技能。

7.3 蛋白质数据库：UniProt

UniProt 是蛋白质序列与功能注释的权威数据库，分为三层：

层	说明
UniProtKB/Swiss-Prot	人工注释、高质量（首选）
UniProtKB/TrEMBL	自动注释、规模大、冗余多
UniProt Reference Proteomes	精选物种的参考蛋白组

蛋白条目有稳定的 **UniProt AC**（如 `P01308` 是人胰岛素）和 **Entry Name**（如 `INS_HUMAN`）。

7.4 结构数据库：PDB

PDB（Protein Data Bank）存放生物大分子的**三维结构**（X 射线、冷冻电镜、核磁等测定的坐标）。

- 每个结构有 4 位字符的 PDB ID（如 `2M21`）；
- 文件格式有 PDB 和更现代的 mmCIF；
- 蛋白结构分析的输入常来自这里。

> **提示** Biopython 的 `Bio.PDB` 模块就是解析 PDB 结构的工具。第 1 章提到蛋白质结构层次，PDB 存的就是三级/四级结构坐标。

7.5 基因组数据库：Ensembl

Ensembl（欧洲）与 **UCSC Genome Browser**（美国）是两大基因组浏览平台，提供：

- 物种参考基因组序列；
- 基因注释（转录本、外显子结构）；
- 变异、调控、比较基因组学数据。

7.6 标识符：数据世界的"身份证"

理解标识符（Identifier）是检索数据的基本功：

类型	例子	说明
GenBank accession	AB000001.2	版本号在点号后
RefSeq	NM_000207.3	前缀表示分子类型
UniProt AC	P01308	蛋白稳定编号
PDB ID	2M21	结构编号
Gene ID / Entrez ID	3639	NCBI 基因编号
Ensembl ID	ENSG00000000003	基因组注释编号

> **注意** 检索时要注意区分"序列标识符"和"基因标识符"——同一个基因在不同数据库里有不同的编号。跨库转换（mapping）是生信日常操作，Biopython 教程第 7 章会教你怎么用 `Entrez` 自动完成。

7.7 小结与练习

小结

- 三大中心：NCBI / EBI / DDBJ，数据同步；
- GenBank 存原始提交，RefSeq 是精选参考库；
- UniProt 管蛋白（Swiss-Prot 质量高），PDB 管结构，Ensembl 管基因组；
- 标识符前缀有规律，学会"看号识类型"。

练习

1. `NP_000198.1` 最可能是什么？（蛋白 / mRNA / 结构）
2. 想查一个蛋白的 3D 结构，应该去哪个数据库？
3. 为什么 RefSeq 通常比 GenBank 更适合做参考序列？

第 8 章 BLAST 与相似性搜索

8.1 BLAST 是什么

BLAST（Basic Local Alignment Search Tool）是 NCBI 提供的**序列相似性搜索工具**：把一条新序列拿去和数据库里亿万条序列比对，找出最相似的记录。

它的常见用途：

- 判断"我这条序列是什么"（物种/基因归属）；
- 找同源序列（功能注释）；
- 发现序列变异。

8.2 工作原理（直觉版）

BLAST 的核心思想是**"种子-扩展"（seed-and-extend）**：

1. 把查询序列切成短片段（"种子"）；
2. 在数据库中快速查找含有相同/相似种子的序列（利用索引加速）；
3. 从种子向两侧扩展比对，直到得分不再上升；
4. 对每条命中计算统计显著性（E-value）。

> **提示** 这个"先找小片段、再扩展"的思路，是 BLAST 比全库两两比对快几个数量级的原因。你不必记住算法细节，但理解"种子-扩展"能帮你读懂结果。

8.3 结果解读：四个关键数值

BLAST 结果页有几个必看指标：

指标	含义	怎么用
Score（得分）	比对打分（位分值）	越高越好
E-value	期望值：**随机**出现同样好结果的概率	越小越显著（如 1e-50）
Identity	一致性百分比	序列有多像
Query Cover	查询序列被覆盖的比例	覆盖率

E-value 是最重要的筛选指标：

- E < 1e-5：通常认为显著同源；
- E 在 0.001~0.05：弱信号，需要谨慎；
- E > 0.1：很可能是随机相似。

> **注意** E-value 与数据库大小有关——数据库越大，同样的比对得分对应的 E 值越大（更不显著）。所以"我 BLAST 出的 E 值小"要在同一数据库背景下比较。

8.4 BLAST 的常用变体

程序	查询	数据库	用途
blastn	核酸	核酸	DNA 对 DNA
blastp	蛋白	蛋白	蛋白对蛋白
blastx	核酸（翻译）	蛋白	用核苷酸查蛋白（如新序列注释）
tblastn	蛋白	核酸（翻译）	蛋白查基因组
megablast	核酸	核酸	高一致、大数据集

> **提示** 记忆技巧：**"n" 结尾查核酸库，"p" 结尾查蛋白库**；带 `x` 表示"查询序列先翻译"。初学者最常用的组合是 `blastn` 和 `blastp`。

8.5 在 Biopython 中怎么用（预览）

Biopython 主要通过两个途径使用 BLAST：

- `Bio.Blast.NCBIWWW`：在线调用 NCBI 的 BLAST 服务；
- `Bio.Blast.NCBIXML`：解析本地 `blast+` 命令行工具输出的 XML。

```python
from Bio.Blast import NCBIWWW

# 在线 BLAST（概念示例，运行时需联网）
result_handle = NCBIWWW.qblast("blastn", "nt", "ATGGCCATTGTAATGGGCCGCTGAAAGGGTGCCCGATAG")
```

> **注意** 这里只是"预览"，完整的参数与结果解析会在 Biopython 教程第 7 章展开。现在你只需知道：**BLAST 结果有标准格式，可以用程序解析**。

8.6 小结与练习

小结

- BLAST = 序列相似性搜索，核心思想是种子-扩展；
- E-value 越小越显著；Identity 看相似程度；Cover 看覆盖范围；
- blastn/blastp/blastx 等变体对应不同"查询×数据库"组合；
- Biopython 可以调用在线 BLAST 并解析结果。

练习

1. E-value 和 Identity 各回答什么问题?
2. 有一条未知 DNA 序列想判断它是否编码蛋白，该用 blastx 还是 blastn?
3. 为什么 E-value 受数据库大小影响?

第 9 章 分子进化与系统发育树

9.1 进化与序列的关系

同一祖先基因在不同物种中不断积累突变，导致序列产生差异。**序列差异越大，通常意味着分化越久**。因此，序列可以当作"进化的化石记录"来推断物种/基因之间的亲缘关系。

9.2 一棵树由什么组成

系统发育树（Phylogenetic tree）用树状图表示演化关系，基本术语：

术语	含义
叶（Leaf/Tip）	现存的物种或序列
内部节点（Node）	共同祖先（推断的）
分支（Branch）	进化关系连线，长度可表示差异量
根（Root）	所有序列的共同祖先
外群（Outgroup）	已知亲缘较远、用于"定根"的物种
分支支持度	如 bootstrap 值，表示该分支的可靠程度

9.3 读树：Newick 格式

树最常见的文本表示是 **Newick 格式**（括号表示法），Biopython 的 `Phylo` 模块解析的就是它：

```
(A:0.1,B:0.2,(C:0.3,D:0.1):0.2);
```

解读：

- `A:0.1`：A 的分支长度为 0.1；
- `(C:0.3,D:0.1):0.2`：C 和 D 先聚成一簇，该簇到根的分支长 0.2；
- 整体表示：A、B 先分离，C、D 关系更近。

再举一例：

```
(人类,(黑猩猩,大猩猩));
```

这棵树表示：黑猩猩和大猩猩的亲缘关系比它们与人类更近（共同祖先更晚）。

> **提示** 练习读 Newick 的好方法是"找最内层括号"：**最内层括号里的两枝关系最近**。掌握这个直觉，任何树都能读。

9.4 建树方法（概念层面）

从多序列比对到一棵树，主流方法有：

方法	思路	特点
邻接法（NJ）	基于距离矩阵，逐步合并最近的两枝	快，适合初步探索
最大简约法（MP）	找"突变次数最少"的树	概念直观，慢
最大似然法（ML）	在进化模型下找"最可能"的树	统计严谨，计算量大
贝叶斯法（Bayesian）	类似 ML，估计后验概率	适合复杂模型

> **注意** 建树是个完整的研究领域。初学阶段只需记住：**先做多序列比对（第 6 章），再算距离/模型，最后建树**。Biopython 的 `Bio.Phylo` 可以读取、绘制、操作现成的树，而 `Bio.Phylo.TreeConstruction` 可以做基础的 NJ 建树。

9.5 小结与练习

小结

- 序列差异反映进化关系；
- 树的要素：叶、节点、分支、根、外群；
- Newick 括号格式是树的通用文本表示，最内层括号关系最近；
- 建树方法：NJ（快）、MP、ML、Bayesian（严谨但慢）。

练习

1. 在 `(A,(B,C));` 中，B 和 C 谁与 A 关系更近？
2. Newick 里分支长度 `:0.5` 可能代表什么？
3. 为什么外群对定根很重要？

第 10 章 为 Biopython 铺路：从概念到工具的映射

10.1 你的知识地图

到这一章，你已经建立了完整的知识框架。把它串起来看：

分子生物学概念		生物信息学概念		Biopython 工具
DNA / RNA / 蛋白质	→	序列（字符串）	→	`Bio.Seq.Seq`
碱基配对、方向	→	互补 / 反向互补	→	`.complement()` `.reverse_complement()`
转录 / 翻译 / 密码子	→	序列转换、蛋白翻译	→	`.transcribe()` `.translate()`
基因 / 注释	→	带注释的记录	→	`Bio.SeqRecord.SeqRecord`
FASTA / FASTQ / GenBank / EMBL	→	序列文件读写	→	`Bio.SeqIO`
Clustal / 比对	→	多序列比对文件	→	`Bio.AlignIO`
GC 含量 / 模体	→	序列统计	→	`Bio.SeqUtils`
比对 / 相似性	→	双序列比对	→	`Bio.Align.PairwiseAligner`
数据库 / 检索	→	NCBI 在线检索	→	`Bio.Entrez`
BLAST	→	相似性搜索	→	`Bio.Blast`
进化树	→	树的结构化表示	→	`Bio.Phylo`
蛋白质理化性质	→	蛋白分析	→	`Bio.SeqUtils.ProtParam`
蛋白质结构	→	结构文件解析	→	`Bio.PDB`
限制酶 / 模体	→	特殊序列工具	→	`Bio.Restriction` / `Bio.motifs`

> **核心观点** 你在本教程学到的每一个概念，在 Biopython 里都有一个对应的模块或对象。**概念越清晰，代码越容易写。**

10.2 一个典型分析流程

把概念连起来，看看一个真实的入门分析流程长什么样：

```
1. 拿到数据（FASTA / FASTQ 文件）          ← 第 4 章
2. 读取序列（SeqIO）                       ← Biopython
3. 质控与清洗（去 N、去短序列）             ← 第 3、5 章
4. 计算统计量（GC、长度分布）               ← 第 5 章
5. 找 ORF / 翻译                           ← 第 2、5 章
6. 注释（BLAST / 数据库）                  ← 第 7、8 章
```

```
7. 多序列比对 + 建树                    ← 第 6、9 章
8. 可视化与报告
```

> **提示** 这也正是 Biopython 教程第 11 章"综合实战"的骨架。你在本教程建立的"流程思维"，会在那里变成完整的可运行项目。

10.3 常见误区清单

误区	正解
把"相似性"说成"同源性"	相似是数值，同源是性质
以为 FASTA 含质量信息	质量在 FASTQ 里
忘记序列有方向	公共数据按 5'→3' 存储
用 `complement()` 当反向互补	另一条链要 `reverse_complement()`
忽视 E-value 只看 Identity	两者都要看
不知道 `NM_`/`NP_` 前缀含义	前缀揭示分子类型
把 GenBank 的 `//` 当成注释	它是记录结束符

10.4 下一步：开始学 Biopython

当你把本教程的练习做完、能用自己的话解释"中心法则""FASTA""比对""E-value"，就可以正式开始《Biopython 零基础实战教程》了。那里会教你：

- 安装 Biopython 并开始第一个脚本；
- 用 `Seq` / `SeqRecord` / `SeqIO` 处理真实数据；
- 用 `PairwiseAligner` 做比对、用 `Entrez` 检索数据库；
- 用 `Phylo` 画进化树、用 `PDB` 分析结构；
- 完成 4 个综合实战项目。

两本教程合在一起，就是一条从零到实战的完整路径。

10.5 小结

- 建立"概念 → 工具"的映射表，是最高效的学习方式；
- 典型流程：读数据 → 质控 → 统计 → 注释 → 比对 → 建树；

- 避免常见误区，能让你少走很多弯路；
- 本教程的终点，就是 Biopython 教程的起点。

附录 A：完整遗传密码表

mRNA 密码子表（U 表示 RNA 中的尿嘧啶；`*` 为终止密码子）：

密码子	氨基酸	密码子	氨基酸	密码子	氨基酸	密码子	氨基酸
UUU	F 苯丙氨酸	UCU	S 丝氨酸	UAU	Y 酪氨酸	UGU	C 半胱氨酸
UUC	F 苯丙氨酸	UCC	S 丝氨酸	UAC	Y 酪氨酸	UGC	C 半胱氨酸
UUA	L 亮氨酸	UCA	S 丝氨酸	UAA	* 终止	UGA	* 终止
UUG	L 亮氨酸	UCG	S 丝氨酸	UAG	* 终止	UGG	W 色氨酸
CUU	L 亮氨酸	CCU	P 脯氨酸	CAU	H 组氨酸	CGU	R 精氨酸
CUC	L 亮氨酸	CCC	P 脯氨酸	CAC	H 组氨酸	CGC	R 精氨酸
CUA	L 亮氨酸	CCA	P 脯氨酸	CAA	Q 谷氨酰胺	CGA	R 精氨酸
CUG	L 亮氨酸	CCG	P 脯氨酸	CAG	Q 谷氨酰胺	CGG	R 精氨酸
AUU	I 异亮氨酸	ACU	T 苏氨酸	AAU	N 天冬酰胺	AGU	S 丝氨酸
AUC	I 异亮氨酸	ACC	T 苏氨酸	AAC	N 天冬酰胺	AGC	S 丝氨酸
AUA	I 异亮氨酸	ACA	T 苏氨酸	AAA	K 赖氨酸	AGA	R 精氨酸
AUG	M 甲硫氨酸*	ACG	T 苏氨酸	AAG	K 赖氨酸	AGG	R 精氨酸
GUU	V 缬氨酸	GCU	A 丙氨酸	GAU	D 天冬氨酸	GGU	G 甘氨酸
GUC	V 缬氨酸	GCC	A 丙氨酸	GAC	D 天冬氨酸	GGC	G 甘氨酸
GUA	V 缬氨酸	GCA	A 丙氨酸	GAA	E 谷氨酸	GGA	G 甘氨酸
GUG	V 缬氨酸	GCG	A 丙氨酸	GAG	E 谷氨酸	GGG	G 甘氨酸

注意 表中 AUG 同时是甲硫氨酸和起始密码子；终止密码子 UAA、UAG、UGA 不编码氨基酸。翻译时遇到终止密码子即结束。

附录 B：术语速查表

英文术语	中文	一句话解释
Nucleotide	核苷酸	DNA/RNA 的基本单元（碱基+糖+磷酸）
Base pair (bp)	碱基对	双链上配对的两个碱基，也作长度单位
Gene	基因	有功能意义的 DNA 片段
Genome	基因组	一个生物体的全部遗传物质
Central Dogma	中心法则	DNA→RNA→蛋白质的信息流动
Transcription	转录	DNA 合成 mRNA（T 变 U）
Translation	翻译	mRNA 按密码子合成蛋白质
Codon	密码子	mRNA 上 3 个碱基一组
Reading Frame	读框	从不同位置开始读密码子
ORF	开放阅读框	从起始到终止的编码片段
Mutation	突变	序列发生改变
SNP	单核苷酸多态性	单个碱基的个体差异
FASTA	FASTA 格式	最简单的序列文件格式
FASTQ	FASTQ 格式	含质量值的测序格式
Phred score	Phred 质量值	碱基可靠性的数值表示
Accession	序列编号	数据库中的稳定标识符
Alignment	比对	对齐序列找相似位点
Similarity	相似性	数值上有多像
Homology	同源性	是否来自共同祖先
Ortholog	直系同源	跨物种的同源基因
Paralog	旁系同源	同物种内复制产生的同源
Gap	缺口	比对中的插入/缺失位置
Substitution matrix	替换矩阵	氨基酸替换打分表（如 BLOSUM62）
MSA	多序列比对	多条序列同时比对
BLAST	BLAST	序列相似性搜索工具

英文术语	中文	一句话解释
E-value	期望值	随机出现该结果的可能性
Phylogenetic tree	系统发育树	表示演化关系的树状图
Newick	Newick 格式	树的括号文本表示
Bootstrap	自举支持度	分支可靠性的统计值

附录 C：推荐学习资源

资源	类型	说明
NCBI	数据库/工具	BLAST、GenBank、PubMed：https://www.ncbi.nlm.nih.gov
UniProt	数据库	蛋白质序列与注释：https://www.uniprot.org
RCSB PDB	数据库	蛋白质三维结构：https://www.rcsb.org
Ensembl	数据库	基因组与注释：https://www.ensembl.org
Biopython 官方文档	文档	https://biopython.org
Biopython 教程（本书配套）	教程	从零到实战的 Python 生物信息学
Rosalind	练习平台	编程+生信小练习：https://rosalind.info
NCBI 教程/学习页	教程	数据库与 BLAST 的官方教学

> **提示** 学习生信最重要的是"动手+查文档"。先会用数据库检索，再会用工具，最后读懂原理——本教程已经把原理部分准备好了。