LECTURE 12 · HANDOUT

Randomness, Text and Files

Every slide from the lecture, with what was said over it. The slides are rendered natively — the text is selectable, searchable, and prints cleanly.



Slide 1 · Randomness, Text and Files

> 本讲把第十一讲的异常和测试用到外部数据上。我们从可复现的随机数据开始，经过字符与字节、文件、正则，完成日志统计与诊断。正文64页，后20页为按需查阅的参考内容，不安排全部讲完。

Review: Failures and Tests 回顾

Failure	Boundary	Evidence
Cannot open a file	file operation	exception + path
Cannot parse a record	one input record	line + raw text + reason
Wrong statistics	program logic	known cases + tests

Slide 2 · Review: Failures and Tests 回顾

> L11区分了崩溃与悄悄算错。本讲把失败边界具体到文件、单条记录和最终统计。请学生指出：程序跑完但少算十条记录，属于哪一类失败？不能只以没有异常为正确依据。

A Log Processing Task 日志处理

A file contains valid records and malformed lines.

INFO user=alice duration=42ms

ERROR user= duration=abc

Deliver counts, a CSV file, and useful diagnostics.

Slide 3 · A Log Processing Task 日志处理

> 先只描述任务，不给代码。日志中有正确记录、缺少用户名和非数字时长。学生需要交统计、可读回的CSV，以及能直接定位到问题输入的说明。问全班：只显示“解析失败”够不够？让他们说出文件名、行号和原始内容。

Prediction: How Many Bytes? 开场题

Predict both numbers. 先预测，编码部分揭晓。

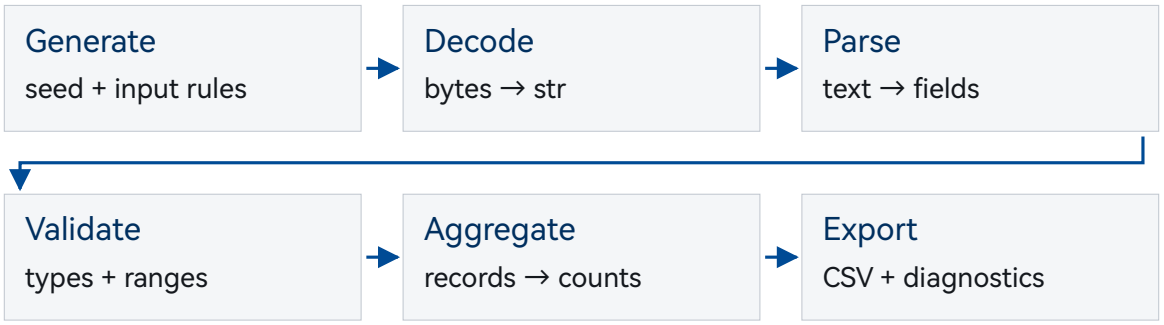
```
s = "A中😀"
print(len(s))
print(len(s.encode("utf-8")))
```

4

Slide 4 · Prediction: How Many Bytes? 开场题

> 先举手选答案：两个长度相同吗？请说出理由，暂不运行、不公布。第22页回到这一页解释，不重复扫码。本题只依赖已经见过的字符串长度；encode暂作为一个待解释操作。

The Data Processing Pipeline 处理流程



5

Slide 5 · The Data Processing Pipeline 处理流程

> 主线由这个任务推进：控制输入生成条件，解释字节，读成文本，解析字段，验证并统计，最后导出。诊断贯穿每个可能失败的边界。后面每完成一段就指出已经具备哪一步能力。

Random Data for Experiments 随机实验



Simulation: sample possible outcomes.

Testing: exercise different inputs.

Debugging: replay the failing experiment.

Slide 6 · Random Data for Experiments 随机实验

> 用同一掷骰任务解释三个需求：结果要有变化、结果空间有约束、出现问题时要重放。这些需求分别引出抽样函数、测试和种子。

Integer Sampling 整数抽样

Expression	Possible values	Upper endpoint
randint(1, 6)	1, 2, 3, 4, 5, 6	included
randrange(1, 6)	1, 2, 3, 4, 5	excluded
randrange(0, 10, 2)	0, 2, 4, 6, 8	follows range

> 先列候选值，再选择抽样函数。

Slide 7 · Integer Sampling 整数抽样

> 让学生先指出模拟骰子应该选哪一行，再说明randrange从range产生的整数中抽一个。右端点规则来自候选集合，不是所有random函数共同的一条规则。

Floating-Point Sampling 浮点抽样

Expression	Range / rule
random()	0.0 ≤ x < 1.0
uniform(a, b)	a + (b − a) × random()
Endpoint b	may occur because of floating-point rounding

Slide 8 · Floating-Point Sampling 浮点抽样

> 先用单位区间解释random，再用一次缩放和平移解释uniform。对uniform不要照搬randrange的右端点结论：实际浮点运算的舍入可能使b出现。Monte Carlo的完整示例在参考页。

Sampling and Replacement 放回与不放回

code

```
1  import random
2  rng = random.Random(42)
3  print(rng.choice(["A", "B", "C"]))
4  print(rng.sample([1, 1, 2], k=3))
5  print(rng.choices(["A", "B"], k=5))
```

output

```
C
[1, 2, 1]
['A', 'A', 'A', 'B', 'B']
```


Slide 9 · Sampling and Replacement 放回与不放回

> choice取一个；sample从总体位置中无放回取k个；choices允许再次抽到同一位置。请注意两个1对应两个位置。即使sample没有重复抽同一位置，结果值仍可重复。先让学生预测三个位置全取时两个1会不会都出现。

In-Place Shuffling 原地洗牌

code

```
1  import random
2  rng = random.Random(1602)
3  original = [1, 2, 3, 4, 5]
4  cards = original.copy()
5  result = rng.shuffle(cards)
6  print(original)
7  print(cards)
8  print(result)
```

output

```
[1, 2, 3, 4, 5]
[4, 1, 2, 5, 3]
None
```

10

Slide 10 · In-Place Shuffling 原地洗牌

> 用L05的原地修改概念讲shuffle：返回None，变化发生在cards对象里。先复制再打乱，就能保留original。请学生指出把cards赋成shuffle返回值会出现什么问题。

Weighted Sampling 带权抽样

code

```
1  import random
2  rng = random.Random(1602)
3  levels = rng.choices(
4      ["INFO", "WARN", "ERROR"],
5      weights=[6, 3, 1], k=100,
6  )
7  for level in ["INFO", "WARN", "ERROR"]:
8      print(level, levels.count(level))
```

output

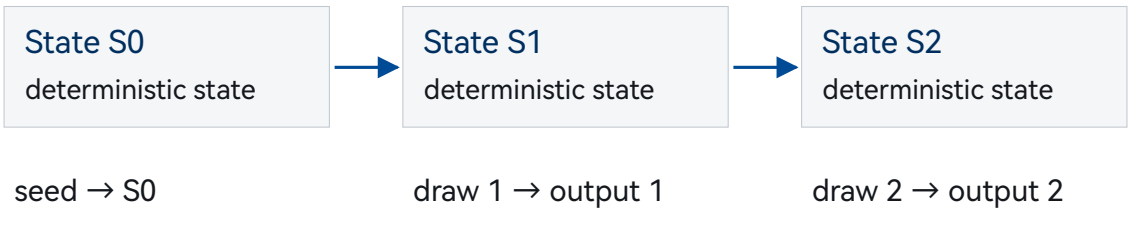
```
INFO 58
WARN 32
ERROR 10
```

11

Slide 11 · Weighted Sampling 带权抽样

> 三个日志级别按6:3:1设置相对权重。输出由当前Python3.13实际运行得到。100条日志不保证恰好60、30、10条；权重规定抽样概率，不规定每批的配额。

Pseudorandom State 伪随机状态



seed → S0

draw 1 → output 1

draw 2 → output 2

Same start + same calls → same sequence

Slide 12 · Pseudorandom State 伪随机状态

> 把随机生成器看成保存状态的对象：一次调用既给出一个数，也改变内部状态。相同初始状态和相同调用过程才能重放。改变中间调用顺序，会改变后面的输出。不同种子也不保证某一次输出必然不同。

Independent Generators 独立随机对象

code

```python
import random
def rolls(seed):
    rng = random.Random(seed)
    return [rng.randint(1, 6) for _ in range(5)]

print(rolls(1602))
print(rolls(1602))
print(rolls(1602) == rolls(1602))
```

output

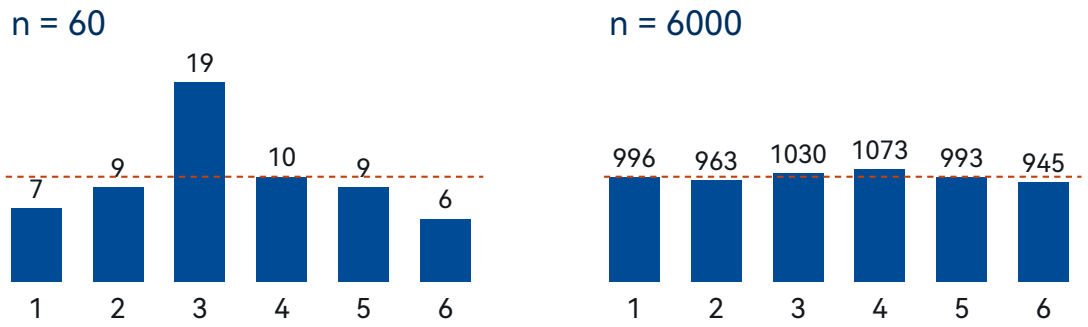
```
[3, 3, 4, 6, 2]
[3, 3, 4, 6, 2]
True
```

13

Slide 13 · Independent Generators 独立随机对象

> Random(seed)是已经学过的对象用法，各实例有独立状态。函数一次建立一个实例，让外部指定种子，便于调试。口头补充反例：若在循环每次抽样前都seed，会一直回到第一个结果；参考代码库有reset示例。

Dice Frequencies 骰子频数



Orange line: expected frequency 1/6

Measured with seed 1602 · 固定条件重放；有限样本仍有波动。

14

Slide 14 · Dice Frequencies 骰子频数

> 数字由固定种子的60次和6000次试验计算，柱顶是实际计数，横线是理论频率六分之一。大样本通常更稳定，但不保证每多投一次误差都下降。这是模拟证据，不是概率定理的证明。

Tests for Random Programs 随机程序测试

- **Range and count**: every roll is in 1…6; exactly n results.
- **State and mutation**: replay the same calls; check the original sequence.
- **Reproduction record**: seed, input, Python version and call sequence.
- **Security**: use `secrets` for tokens and passwords.

Slide 15 · Tests for Random Programs 随机程序测试

> 随机程序不必只测试某个固定答案。先测范围、长度和修改行为，再测固定条件重放。跨版本的高层抽样算法可能变化，所以记录版本。random不适合安全用途。段末转向保存数据：刚才的结果写到文件时，保存的是字符还是字节?

Written Symbols and Digital Text 文字表示



People recognize **visible shapes**.

Programs exchange **agreed representations**.

字形怎样变成可交换的文本?

Slide 16 · Written Symbols and Digital Text 文字表示

> 给原图留出观看时间。这张甲骨照片只承担一个问题：人看到形状，程序要怎样存储和交换文字？不猜测出土地、年代或具体释文。下一页从图形变化走向字符身份，不把照片当成编码数据本身。

Glyphs and Character Identity 字形与字符



Glyph: a visible shape in a writing style or font.

Character identity: what a text standard represents.

字形不同，不必然是不同字符；历史字形也不能直接等同码点。

Slide 17 · Glyphs and Character Identity 字形与字符

> 原图的古今字形帮助区分看到什么与表示什么。不要说每一对古今字形必有同一码点。用现代字体中同一个A的不同形状作熟悉类比：文本标准需要处理字符身份，而不是给每张图分配一个编号。

ASCII and Unicode 字符编码标准



- **ASCII**: 128 assigned values.
- **Unicode code point 码点**: a number such as U+4E2D.
- **Encoding 编码**: how text is represented as bytes.

> 码点与文件字节要分开讨论。

Slide 18 · ASCII and Unicode 字符编码标准

> L02讲过ASCII。这里先扩大字符表示范围，再把码点与字节分开。Unicode不仅是编号表，还定义字符属性与相关规则；本课聚焦码点。不背某版本收录字符总数。

Character Lookup 字符查询

code

```
for ch in "A中😀":
    print(ch, ord(ch), f"U+{ord(ch):04X}")
print(chr(20013) == "\u4e2d")
```

output

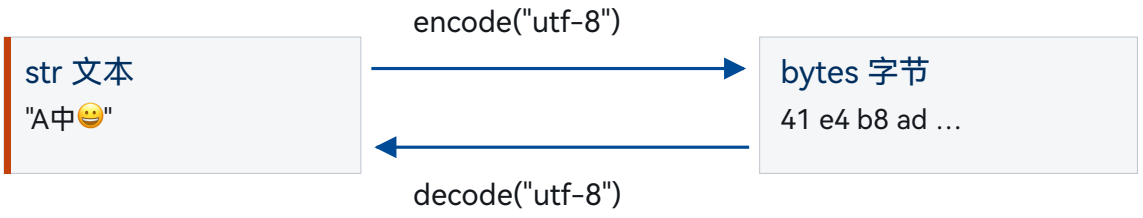
```
A 65 U+0041
中 20013 U+4E2D
😀 128512 U+1F600
True
```

19

Slide 19 · Character Lookup 字符查询

> ord取单个字符的码点，chr做反向转换。U+是码点记法，字符串里的反斜杠u是Python转义写法，两者都不说明文件占几个字节。通过中对应20013把抽象编号落到具体例子。

Text and Byte Sequences 文本与字节



20

Slide 20 · Text and Byte Sequences 文本与字节

> 方向是本页核心：文本编码成字节，字节解码回文本。bytes的索引是0到255的整数。字节对象没有一个能让程序无条件确定原始编码的标签；必须依靠协议、文件来源或显式约定。

UTF-8 Byte Lengths 字节长度

A	中	😀
U+0041	U+4E2D	U+1F600
41	e4 b8 ad	f0 9f 98 80

3 code points → 1 + 3 + 4 = 8 bytes

21

Slide 21 · UTF-8 Byte Lengths 字节长度

> 三个码点实际编码为1、3、4个字节，分组必须与实跑hex一致。不要概括所有中文都是三字节、所有emoji都是四字节：罕见汉字可在补充平面，视觉emoji也可能是码点序列。码点范围表放讲义参考。

The Opening Prediction 开场题回顾

code

```
1  s = "A中😀"
2  print(len(s))
3  print(len(s.encode("utf-8")))
```

output

```
3
8
```

22

Slide 22 · The Opening Prediction 开场题回顾

> 回收第4页的两个数字：3和8。len(str)数码点，len(bytes)数字节。请先让原来认为相同的同学用字节块重新解释。不要重复发同一道题。组合字符的视觉长度边界留参考页。

Decoding and Mojibake 解码与乱码

code

```
1  data = "你好".encode("utf-8")
2  print(data.decode("utf-8"))
3  wrong = data.decode("latin-1")
4  print(repr(wrong))
5  print(wrong.encode("latin-1") == data)
```

output

```
你好
'ä½\xa0å¥½'
True
```

> 不报错，也需要核对解码后的内容。

Slide 23 · Decoding and Mojibake 解码与乱码

同样的字节，用UTF-8能回到你好，用Latin-1会得到错误文本但不报错。这里还可逆，因为没有丢字节；不能一概说所有乱码无法恢复。下一类失败是用ASCII解码抛异常，完整异常见讲义和R19。

Source and File Encodings 两种解码边界

Object	Who decodes it?	Rule
Python source (.py)	Python interpreter	UTF-8 by default in Python 3
Input data (.log)	your file-reading code	use the actual source encoding
New course text	the program that writes it	choose UTF-8 explicitly

Slide 24 · Source and File Encodings 两种解码边界

编辑器把.py存成UTF-8，不能保证程序打开的每个外部文件都是UTF-8。画清源文件到解释器、数据文件到open两个边界。新建数据约定UTF-8，已有数据根据真实编码读取。

Encoding Diagnostics 编码诊断

- **Locate**: file name, byte offset and the actual error.
- **Check the source**: declared format and writing program.
- **Decode again** with the correct encoding; check known content.
- **Avoid silent loss**: `ignore` and `replace` change the data.

25

Slide 25 · Encoding Diagnostics 编码诊断

> 诊断要保留失败上下文。UnicodeDecodeError中的start/end是相关解码输入对象的偏移，不在这里承诺它总是整个大文件的绝对偏移。先核对来源，再重读；不要靠忽略错误把程序改成看似成功。

Quiz 1 · Randomness and Encoding

扫码作答 · 3 题

1 随机状态与可复现性　2 不放回抽样与重复值
3 乱码与解码失败

根据刚才学过的规则推理；本轮只提交一次。

https://taohuang.info/cs1602/zh/quiz/12/

26

Slide 26 · Quiz 1 · Randomness and Encoding

> 第一轮三题，给学生约三分钟思考，留一分钟解释。Q1同种子同调用，Q2sample可以包含重复值，Q3Latin-1不报错不证明编码正确。所有概念此前已教，不引入新API。

File Processing 文件处理

> 到这里已经知道字节怎样解释为文本。下面处理找到文件、打开文件、读取与写入的实际步骤。把文件级错误与记录级错误区分开，为正则解析做好准备。



Slide 28 · Paths and the Working Directory 工作目录

> 目录树明确脚本在src，数据在data。从project运行和从src运行，相对路径结果不同。FileNotFoundError不一定是文件没下载，也可能是工作目录不符。固定脚本位置找资源可用__file__，放参考说明，不改变工作目录来掩盖问题。

Paths with pathlib 路径对象

code

```python
from pathlib import Path
root = Path("lab12")
root.mkdir(exist_ok=True)
p = root / "app.log"
p.write_text("hello\n", encoding="utf-8")
print(p.name, p.suffix, p.parent)
print(p.exists(), p.stat().st_size)
```

output

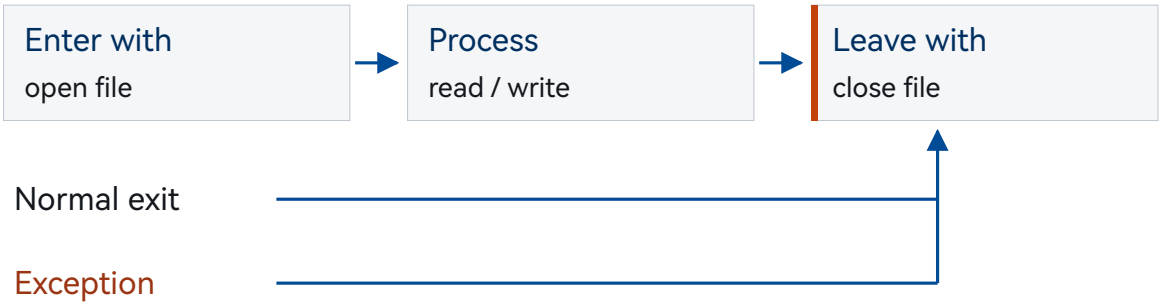
```
app.log .log lab12
True 6
```

29

Slide 29 · Paths with pathlib 路径对象

所有例子在独立临时工作目录运行，但学生代码使用相对的lab12练习目录，适用于Windows和macOS。斜杠是Path的运算符，name、suffix、parent描述结构。exists只是一次观察，打开时仍需处理失败。

File Lifetime with with 文件生命周期



30

Slide 30 · File Lifetime with with 文件生命周期

with负责离开代码块时关闭文件，包括正常离开和有异常离开；它不负责吞掉异常。open本身失败时没有成功进入文件上下文。联系L11资源清理，不展开__enter__/__exit__实现。

File Modes 文件模式

Mode	File missing	File exists
r	FileNotFoundError	read from start
w	create	truncate immediately
a	create	append writes
x	create	FileExistsError

> 覆盖、追加、拒绝覆盖，是三个不同需求。

Slide 31 · File Modes 文件模式

父目录不存在时创建文件也会失败，这张表只比较目标文件在可用目录中的情况。r默认文本；b选择二进制。问想保留旧文件并拒绝覆盖应该选哪一种。

File Truncation 文件截断

code

```python
from pathlib import Path
p = Path("demo.txt")
p.write_text("keep me", encoding="utf-8")
with p.open("w", encoding="utf-8"):
    pass
print(repr(p.read_text(encoding="utf-8")))
```

output

```
''
```


Slide 32 · File Truncation 文件截断

不要拿真实文件演示。示例创建demo.txt，再用w打开、pass、关闭，内容变空。必须让学生说清发生时间是成功打开时，而非第一次write。文件模式是需要在操作前确认的条件。

Writing Text and Newlines 写入与换行

code

```
with open("demo.txt", "w", encoding="utf-8") as f:
    f.write("one")
    f.writelines(["two", "three"])
with open("demo.txt", encoding="utf-8") as f:
    print(repr(f.read()))
```

output

```
'onetwothree'
```

33

Slide 33 · Writing Text and Newlines 写入与换行

> write与writelines都不自动加换行，后者只是依次写序列中每个字符串。让学生先预测最后文件是三行还是一个连续字符串。换行符要由数据或代码明确写入。

File Reading Methods 文件读取

code

```
import io
text = "A中\nlast"
print(repr(io.StringIO(text).read(2)))
print(repr(io.StringIO(text).readline()))
print(io.StringIO(text).readlines())
```

output

```
'A中'
'A中\n'
['A中\n', 'last']
```

34

Slide 34 · File Reading Methods 文件读取

> StringIO把已知文本当文件流，便于在一页独立展示三种读取。read(2)得到两个字符A和中；若二进制模式read(2)则是至多两个字节。readline保留换行，readlines一次保存全部行。

Blank Lines and EOF 空行与文件结束

code

```
1  import io
2  f = io.StringIO("\nlast")
3  for _ in range(3):
4      print(repr(f.readline()))
```

output

```
'\n'
'last'
''
```

35

Slide 35 · Blank Lines and EOF 空行与文件结束

> 保留往年对文件结尾的辨析。第一行空行返回反斜杠n，最后没有换行的last仍是一条有效行，之后才返回空字符串。请学生解释为什么if not line只适合测试原始readline的EOF，不能在strip后区分空行。

Streaming and Accumulation 流式读取

Program	Retained data	Memory depends on
for line in f: update counts	current line + counts	longest line, buffers, keys
for line in f: lines.append(line)	every line	total stored input
content = f.read()	one complete string	decoded text size

36

Slide 36 · Streaming and Accumulation 流式读取

> 流式读取是一种处理方式，不是内存恒定的承诺。若统计键无限增加，字典也会增长；若每行append到列表，就是保存全量数据。为L13提出按需返回记录的问题，但本讲不用yield。

Whitespace Preservation 空白保留

code

```
line = "  alice  \n"
print(repr(line.strip()))
print(repr(line.rstrip("\n")))
```

output

```
'alice'
'  alice  '
```

37

Slide 37 · Whitespace Preservation 空白保留

> 两个输出实跑对照：strip连用户名前后有效空格也删掉，rstrip换行只删指定字符。默认文本模式已经把常见行尾转为换行；如果业务允许保留其他末尾空白，不要一概清除。

File and Record Errors 失败边界

Scope	Examples	Response
File	missing / permission / decoding	report context; stop this file
Record	missing field / invalid value	report the row; follow policy
Program	unexpected bug	preserve the traceback; fix it

38

Slide 38 · File and Record Errors 失败边界

> 不要用一层except Exception把三类失败都当成坏数据。哪些记录允许跳过由任务契约决定。文件无法解码时不能声称后续行都已安全处理；异常的边界和恢复策略要明确。

Line Numbers and Raw Input 行号与原文

```python
with open("app.log", encoding="utf-8") as f:
    for lineno, raw in enumerate(f, start=1):
        line = raw.rstrip("\n")
        # Parse one record; keep lineno and raw.
        print(lineno, repr(line))
```


Slide 39 · Line Numbers and Raw Input 行号与原文

> 这是完整的读取代码，运行前需要准备app.log，因此本页让学生观察代码，不自动执行。enumerate把物理行号固定下来，原文要在任何strip或转换之前保留。完整可运行案例在讲义，不在一页里塞整个解析器。

CSV Quoting 字段中的逗号

code

```python
import csv, io
text = '"Li, Jr.",92\n'
print(text.rstrip("\n").split(","))
print(next(csv.reader(io.StringIO(text))))
```

output

```
['"Li', ' Jr."', '92']
['Li, Jr.', '92']
```


Slide 40 · CSV Quoting 字段中的逗号

> 名字Li逗号Jr.是一个字段，split把它误拆成两段。csv.reader按格式规则识别引号。这里的困难是结构，不是分隔符字符数量；不用正则修补CSV。

CSV Reading and Writing 读写闭环



code

```
import csv
rows = [["name", "score"], ["Li, Jr.", "92"]]
with open("scores.csv", "w",
          encoding="utf-8", newline="") as f:
    csv.writer(f).writerows(rows)
with open("scores.csv", encoding="utf-8",
          newline="") as f:
    loaded = list(csv.reader(f))
print(loaded == rows)
```

output

```
True
```

41

Slide 41 · CSV Reading and Writing 读写闭环

> 写入再读取比较整张表；这里提前用字符串表示分数，让往返类型一致。CSV读写都使用newline空字符串，让模块管理换行；encoding控制文本编码。表头也是显式写入的一行。

CSV Headers and Field Types 表头与类型



code

```
import csv, io
data = 'name,score\n"Li, Jr.",92\n'
row = next(csv.DictReader(io.StringIO(data)))
print(row["name"])
print(repr(row["score"]))
print(int(row["score"]) + 1)
```

output

```
Li, Jr.
'92'
93
```

42

Slide 42 · CSV Headers and Field Types 表头与类型

> DictReader把表头映射成键，但分数仍是字符串，必须按业务规则转换和检查范围。结构正确、类型正确、取值合法是三步，不混为一谈。问空分数或超过100分应该在哪一步拒绝。

File Processing Boundaries 文件处理小结

- **Locate and decode**: path, working directory, encoding.
- **Manage resources**: mode, `with`, reading strategy.
- **Choose the parser**: CSV library for CSV; string methods for simple separators.
- **Next**: patterns for semi-structured log fields.

Slide 43 · File Processing Boundaries 文件处理小结

> 把本段能力接回主线：文件可以读进来，不代表字段已经可信。CSV有现成格式解析器；普通split能处理很多简单场景，不能为了引出正则而夸大字符串方法的缺陷。

Regular Expressions in Text Tools 正则表达式



Ken Thompson implemented pattern matching in QED.

A text editor needs to find and replace **patterns**, not just fixed words.

His UTF-8 work with Rob Pike connects text tools and representation.

Slide 44 · Regular Expressions in Text Tools 正则表达式

> 讲一个完整的一分钟故事。文本编辑器需要查找某种形状的文本，Thompson在QED中实现模式匹配，这是实用需求推动语言设计的例子。他与Pike的UTF-8工作又呼应前半讲。依据Ritchie的QED历史与Pike亲历邮件，不声称Thompson发明了正则理论，也不展开自动机。

String Methods and Patterns 工具选择

Task	Suitable tool
Separate words with varying whitespace	str.split()
Separate one fixed key=value pair	str.split("=", 1)
Parse quoted comma-separated records	csv
Extract fields with a defined text pattern	re

Slide 45 · String Methods and Patterns 工具选择

> 纠正旧讲义“多一个空格split就坏”的说法，split不带分隔符能处理连续空白。用固定结构与变化模式的差别引出正则。强大工具本身也有调试成本，应选最简单且满足契约的工具。

Raw Strings and Literals 原始字符串

code

```
1  import re
2  text = "v1.2"
3  print(re.findall(r".", text))
4  print(re.findall(r"\.", text))
```

output

```
['v', '1', '.', '2']
['.']
```


Slide 46 · Raw Strings and Literals 原始字符串

> 先分开Python字符串这一层和正则语法这一层。raw string减少Python先处理反斜杠的干扰，但不会取消正则的转义规则。点匹配任意非换行字符，反斜杠点匹配真正点号。

Character Classes 字符集合



code

```
import re
for text in ["123", "１２３", "abc", "中"]:
    digit = bool(re.fullmatch(r"\d+", text))
    ascii_ok = bool(re.fullmatch("[0-9]+", text))
    print(text, digit, ascii_ok)
```

output

```
123 True True
１２３ True False
abc False False
中 False False
```

\w: Unicode alphanumeric or underscore · \s: whitespace · [^...]: complement

Slide 47 · Character Classes 字符集合

> 逐行预测：全角数字满足Unicode十进制数字类，但不属于ASCII的0到9。用户名若要求ASCII小写应明确写a-z，不把反斜杠w等同于英文字母。反斜杠s表示空白，方括号起始的脱字符表示补集。

Quantifiers 重复次数



code

```
1  import re
2  for text in ["", "a", "aaaa"]:
3      plus = re.fullmatch(r"a+", text) is not None
4      star = re.fullmatch(r"a*", text) is not None
5      print(repr(text), plus, star)
```

output

```
'' False True
'a' True True
'aaaa' True True
```

?: 0 or 1 · {n}: exactly n · {m,n}: from m to n

Slide 48 · Quantifiers 重复次数

> 让学生先判断空串：星号允许零次，加号要求至少一次。问号允许零或一次，花括号明确次数。给出a?、a{3}、a{2,4}让学生口头判断。这里用简单a避免同时学字符类与量词；日志中的空用户名用加号拒绝。

Search, Prefix and Full Match 匹配范围

abc12345

search	scan → "123"	[0-9]{3}
match	start only → None	
fullmatch	whole input → None	

Slide 49 · Search, Prefix and Full Match 匹配范围

> 三种函数在相同输入上有不同搜索范围。search在中间发现123就成功，match从开头试，fullmatch要求整个输入符合。整串恰好三位ASCII数字用fullmatch，不把search成功当成验证通过。

Missing Matches 未找到匹配



code

```python
import re
text = "no digits here"
m = re.search(r"[0-9]+", text)
if m is None:
    print("no match:", repr(text))
else:
    print(m.group())
```

output

```
no match: 'no digits here'
```


Slide 50 · Missing Matches 未找到匹配

> 先判断None，再取group。强调未匹配是API的正常结果，不是立刻抛异常。若直接对None取group才会产生AttributeError，完整反例放R19。这个分支是解析器里正常输入与诊断路径的分岔点。

Capturing Groups 捕获分组



code

```
import re
p = (r"user=(?P<user>[a-z]+)"
     r" ms=(?P<ms>[0-9]+)")
m = re.fullmatch(p, "user=alice ms=42")
if m is not None:
    print(m.group(0))
    print(m.group("user"))
    print(m.group("ms"))
    print(m.groupdict())
```

output

```
user=alice ms=42
alice
42
{'user': 'alice', 'ms': '42'}
```


Slide 51 · Capturing Groups 捕获分组

> 把整体与两个命名字段对应起来。group零号返回整体，命名组取字段，groupdict返回名字到字符串的字典。时长42仍是字符串，后续还要转换。让学生指出输出中哪个值的类型还不是int。

Log Pattern Construction 日志模式

```python
import re
p = re.compile(
    r"(?P<level>INFO|WARN|ERROR)\s+"
    r"user=(?P<user>[a-z]+)\s+"
    r"ms=(?P<ms>[0-9]+)"
)
for line in ["INFO user=alice ms=42", "INFO user= ms=x"]:
    m = p.fullmatch(line)
    print(m.groupdict() if m else None)
```


Slide 52 · Log Pattern Construction 日志模式

> 逐段指着输入解释级别、空白、user、时长三个部分。一次只增加一个片段；不要让学生把整个模式当成神秘字符串背下来。示例只有level、user和ms三个字段，与实验的六字段完整日志有区别，不直接泄露整道答案。

Syntax and Semantic Validation 格式与含义

```python
import re
from datetime import date
s = "2026-02-31"
p = r"[0-9]{4}-[0-9]{2}-[0-9]{2}"
print(bool(re.fullmatch(p, s)))
try:
    date.fromisoformat(s)
except ValueError as exc:
    print(type(exc).__name__)
```

output

```
True
ValueError
```

53

Slide 53 · Syntax and Semantic Validation 格式与含义

> 2026年2月31日长得像日期，却不是真实日期。模式负责形状，datetime负责日历语义。这里仅示范标准库验证，不要求学生本节手写日历规则。compile把模式命名复用，不能宣称绕过所有模块函数的缓存而必然更快。

Parsing and Aggregation 解析与统计

- **Read** one record and keep its source location.
- **Parse and validate** before modifying the result.
- **Count** only records accepted by the contract.
- **Report** rejected records with the raw input and reason.

54

Slide 54 · Parsing and Aggregation 解析与统计

> 先解析、后转换、最后更新结果。不要在int转换成功前就setdefault留下空字典。主流程调用独立parse_line，错误分支返回诊断；这样测试可以先测单行，再测整个文件。

Actionable Parse Diagnostics 可定位的诊断

Actual output 实跑输出

```
app.log:2: invalid record
  input='ERROR user='
  expected: INFO|ERROR user=<a-z+>
{'INFO': 1, 'ERROR': 1}
```

- **Location**: file and physical line.
- **Input**: the original record.
- **Reason**: what was expected.
- **Recovery**: valid rows still count.

Slide 55 · Actionable Parse Diagnostics 可定位的诊断

> 本页输出由完整示例实跑生成，显示失败记录的文件名、行号、原文和预期格式，然后继续统计正确记录。学生应该能根据这份输出直接写出一个回归测试。不要只展示“invalid record”一行。

Greedy and Lazy Quantifiers 贪婪与非贪婪



code

```
1  import re
2  s = 'user="alice" action="login"'
3  print(re.findall(r'".*"', s))
4  print(re.findall(r'".*?"', s))
5  print(r'"[^"\n]*"', s))
```

output

```
['"alice" action="login"']
['"alice"', '"login"']
['"alice"', '"login"']
```

56

Slide 56 · Greedy and Lazy Quantifiers 贪婪与非贪婪

> 用同一行两个引号字段比较跨度。贪婪先尽量长，非贪婪先尽量短，但两者都必须满足模式剩余部分。第三种明确排除引号和换行，常常更符合字段边界。不要拿这个例子去承诺能解析任意嵌套HTML。

Replacement and Pattern Splitting 替换与切分



code

```
1  import re
2  s = "user=alice; duration=42ms"
3  p = r"user=[a-z]+"
4  print(re.sub(p, "user=REDACTED", s))
5  print(re.split(r"[;,]\s*", "a, b;c"))
6  print(s)
```

output

```
user=REDACTED; duration=42ms
['a', 'b', 'c']
user=alice; duration=42ms
```

57

Slide 57 · Replacement and Pattern Splitting 替换与切分

> sub返回新字符串，split返回列表。原字符串没有被修改。遮掉用户名只是课堂例子；若要可靠脱敏还需清楚定义所有身份字段，不用本例声称已完成隐私保护。

Quiz 2 · Files and Text Patterns

扫码作答 · 3 题

1 空行、末行与 EOF
2 局部匹配与整串验证
3 命名分组及 None 分支

根据刚才学过的规则推理；本轮只提交一次。

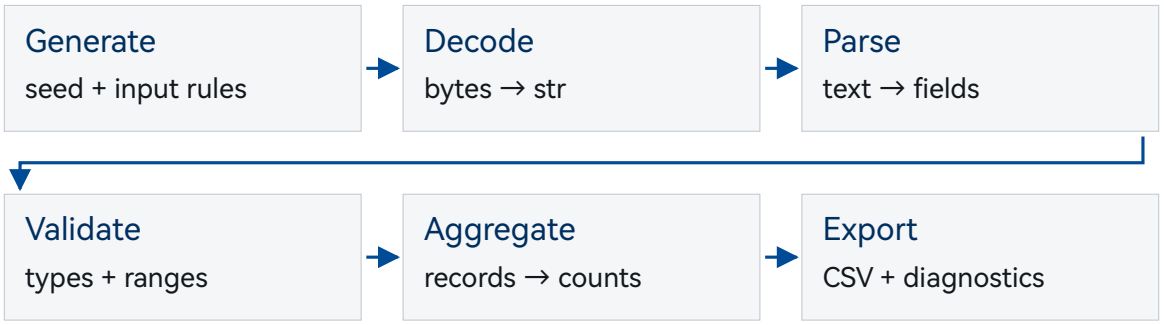
https://taohuang.info/cs1602/zh/quiz/12/

58

Slide 58 · Quiz 2 · Files and Text Patterns

第二轮三题，依次核对空行不等于EOF、search不等于验证、未匹配先处理None。解释Q6时不额外引入新语法。两轮总共六题，开场不重复发题。

From Log File to CSV 综合处理



59

Slide 59 · From Log File to CSV 综合处理

重新走开场流程，一条正常记录经解析和验证进入计数与CSV，一条坏记录转向诊断。强调结果文件与诊断文件应分清，不把报错混进CSV。这里已具备完成Lab12日志主问题的全部基础。

Tests for Text Pipelines 文本处理测试

Case	What to check
Empty / malformed record	no silent loss; stable classification
Last line without newline	the record is still processed
Unicode / comma in field	correct text and CSV round trip
Export and read back	field values and order, not only row count

60

Slide 60 · Tests for Text Pipelines 文本处理测试

> 从L11的测试思想收束。往返只比较条数不够，两条不同记录也可能有同样数量。比较解析后的字段与顺序，而非要求CSV字节完全相同，因为换行与合法引号写法可能不同。

The Log Processing Result 日志处理结果

Deliverable	Evidence
Statistics	accepted records follow the counting contract
CSV	read-back records equal exported records
Diagnostics	path + line + raw input + reason
Accounting	every input line is classified

61

Slide 61 · The Log Processing Result 日志处理结果

> 正面回答开场任务。不再只说“运行成功”。若区分空行、注释、坏行和成功记录，应有完备分类；合计等于读取的物理行数。具体分组由实验契约确定，不能默默忽略一类输入。

Summary 本讲小结

- **Randomness**: control the initial state and the experiment.
- **Encoding**: distinguish code points from stored bytes.
- **Files**: make modes, resources and memory use explicit.
- **Text patterns**: parse, validate, test and diagnose.

Slide 62 · Summary 本讲小结

> 请学生用自己语言复述四个判断，不要求背全套函数名。随机要能重放；文本要分清码点与字节；文件要知道模式和资源边界；解析要配验证和定位信息。四段围绕同一数据流程，而不是四个无关的标准库清单。

Lab 12 and Solo Assignment 3 实验预告

Part	Task
A · Fundamentals	encoding, files, fields, extraction, seeded rolls
B · One log pipeline	load → diagnose → aggregate → CSV → verify
C · Computer skills	SSH login and scp file transfer
Solo assignment 3	big integers over strings · weeks 12–13

Slide 63 · Lab 12 and Solo Assignment 3 实验预告

> 理论与实验是同一天。A做基础题，B围绕同一日志递进，C使用课程指定远程练习环境。个人作业3在第12到13周开展，不再复制旧的项目B本周日截止与口头验收说法。实际截止日期由课程日历和作业配置显示。

Preview: Iterables and Generators 下讲预告

- **Today**: process a file one line at a time.
- **Limitation**: returning a full list still stores all records.
- **L13**: produce and connect records on demand.
- How can a function provide the next record only when needed?

Slide 64 · Preview: Iterables and Generators 下讲预告

> 把L12流式读取的限制作为L13入口。列表返回值会保存全部记录，下一讲讨论按需产生值以及如何组合处理步骤。这里不提前写生成器表达式或yield，先让学生带着明确问题离开。

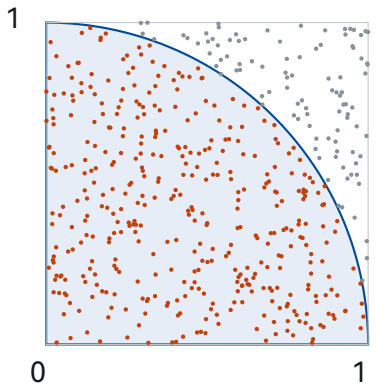
Random API Reference 随机接口参考

API	Rule
randint / randrange	closed interval / choose from range
random / uniform / gauss	unit interval / scaled interval / normal model
choice / sample / choices	one / without / with replacement
shuffle	mutates the sequence; returns None
sample population	sequence required; repeated values allowed

Slide 65 · Random API Reference 随机接口参考

参考页不逐条背。gauss只提示存在正态分布模型，不扩展概率课。Python3.13的sample要求序列，旧幻灯片写可以直接传集合已不适用。空总体和样本数量过大时分别查看对应异常。

Monte Carlo Estimation of Pi 蒙特卡洛



First 500 of 10,000 points · 前500个采样点

Code · quarter-circle sampling

```
import random
rng = random.Random(1602)
inside = 0
n = 10000
for _ in range(n):
    x, y = rng.random(), rng.random()
    if x*x + y*y <= 1:
        inside += 1
```

π ≈ 4 × inside / n

pi estimate: 3.1428

Slide 66 · Monte Carlo Estimation of Pi 蒙特卡洛

在单位正方形中均匀采样，用四分之一圆内比例估计圆周率。图中是相同种子与调用过程的前500个点，估计值由全部一万个点计算。圆内为橙色，外部为灰色。固定条件让实验可重复，不保证误差单调减小。

Birthday Collision Simulation 生日模拟

- **Model**: independent, uniform birthdays on 365 days.
- **Trial**: compare the number of birthdays and distinct birthdays.
- **Repeat**: collisions / trials; set the seed once per experiment.
- **Interpretation**: a finite simulation may cross 50% at a different n.

Slide 67 · Birthday Collision Simulation 生日模拟

旧Lab12生日问题保留为扩展。23人是均匀独立模型下理论概率首次超过一半的人数，有限模拟有波动，不能要求所有随机运行都给这个阈值。评价可复现性和估计方法，不拿一个未经定义的阈值硬判正确。

Bytes and Hexadecimal 二进制序列

code

```
1  data = bytes.fromhex("41 e4 b8 ad")
2  print(data.decode("utf-8"))
3  print(data.hex(" "))
4  editable = bytearray(data)
5  editable[0] = 66
6  print(editable.decode("utf-8"))
```

output

```
A中
41 e4 b8 ad
B中
```


Slide 68 · Bytes and Hexadecimal 二进制序列

bytes不可变，bytearray可变；索引表示单字节整数。hex与fromhex让二进制数据便于检查，不是另一种文本字符编码。把第一个ASCII字节从A改为B，后面的UTF-8字节保持不变。

Unicode Escapes and Names 转义与字符名



code

```
1  import unicodedata as ud
2  print("\x41", "\u4e2d", "\U0001f600")
3  print(ud.name("中"))
4  print(ud.lookup("EURO SIGN"))
5  print("\N{EURO SIGN}")
```

output

```
A 中 😀
CJK UNIFIED IDEOGRAPH-4E2D
€
€
```

69

Slide 69 · Unicode Escapes and Names 转义与字符名

区分Python转义、Unicode码点与编码字节。反斜杠x在str字面量里表示U+00xx的字符，不应总说成一个存储字节。unicodedata通过标准字符名查找，旧课件的同类例子重跑后展示。

UTF-16, UTF-32 and BOM 编码参考



code

```
1  for text in ["A", "中", "😀"]:
2      print(text, len(text.encode("utf-8")),
3            len(text.encode("utf-16-le")),
4            len(text.encode("utf-32-le")))
5  print("A".encode("utf-16").hex(" "))
```

output

```
A 1 2 4
中 3 2 4
😀 4 4 4
ff fe 41 00
```

70

Slide 70 · UTF-16, UTF-32 and BOM 编码参考

前三列使用明确小端编码，不含自动BOM，所以能观察1/3/4与2/2/4等码元长度。UTF-16的补充平面码点需要两个16位码元。最后一行默认UTF-16展示本平台BOM，不把本机字节序当成全部平台固定规则。

Code Points and Displayed Characters 视觉字符



code

```
1  import unicodedata as ud
2  for s in ["é", "e\u0301", "👩‍💻"]:
3      print(repr(s), len(s), len(s.encode("utf-8")))
4  print(ud.normalize("NFC", "e\u0301") == "é")
```

output

```
'é' 1 2
'é' 2 3
'👩\u200d💻' 3 11
True
```

71

Slide 71 · Code Points and Displayed Characters 视觉字符

> é可以是单一码点，也可以是e与组合重音；视觉emoji可以由多个码点组成。len(str)不能当作一般视觉字符数。NFC只用一个真实例子展示规范化意义，不要求学生实现字素簇算法。

Source File Encoding 源码编码



- **Python 3 source** defaults to UTF-8.
- **An encoding declaration** can appear on the first or second line.
- **Editor settings** describe the file being edited.
- **Data files** still need their own decoding rule.

72

Slide 72 · Source File Encoding 源码编码

> 保留旧源码编码页的知识点，避免继续使用Python3.9状态栏截图。若需要展示编辑器状态，使用实际环境的新截图；本页用边界说明已足够，不为截图而截图。读取外部GBK文件仍需正确指定其编码。

Command-Line Arguments 命令行参数

Run: python analyze.py "data folder/app.log"

```python
import sys

print(sys.argv[0])
print(sys.argv[1:])
```


Slide 73 · Command-Line Arguments 命令行参数

在终端运行python analyze.py "data folder/app.log"，脚本名在索引零，路径作为一个参数。输入路径与运行目录要分别解释。命令中的引号由shell处理，不会成为参数值的一部分。

Runtime and Standard Streams 运行环境

Object	Meaning
sys.version_info	interpreter version
sys.argv / sys.path	program arguments / module search path
stdin / stdout / stderr	input / normal output / diagnostics
sys.exit()	raise SystemExit
sys.maxsize	platform Py_ssize_t limit; not the int limit

Slide 74 · Runtime and Standard Streams 运行环境

保留旧sys内容并修正两处误导：sys.path不是数据文件搜索目录，maxsize不是Python整数最大值。标准流可以连接终端、文件或管道，不能说标准流本身就是管道。正常数据与诊断分开有助于重定向。

Filesystem Operations 文件系统操作

code

```python
from pathlib import Path
p = Path("practice")
p.mkdir(exist_ok=True)
(p / "demo.txt").write_text("hello", encoding="utf-8")
for item in sorted(p.iterdir()):
    print(item.name, item.is_file())
```

output

```
demo.txt True
```

75

Slide 75 · Filesystem Operations 文件系统操作

> pathlib面向对象地表达路径，os提供环境变量与系统接口。这里示范创建专用practice目录并枚举，不做破坏性全盘演示。PurePath只处理路径计算，Path还能访问文件系统。Linux或WSL是工具路线，不强制学生装虚拟机。

File Methods and Modes 文件接口参考

Operation	Meaning
read(n) / read()	at most n units / remaining content
readline() / readlines()	one line / list of remaining lines
write(s) / writelines(parts)	write text; no automatic newline
closed / mode / name	file state / mode / name
b / + / a / x	binary / update / append / exclusive create

76

Slide 76 · File Methods and Modes 文件接口参考

> 查表时要同时看对象是文本流还是二进制流。文本read的单位是字符，二进制是字节。追加模式与加号更新模式不能混称。实际读取位置随调用推进，再读不会自动回到开头。

Binary File Copying 二进制复制



code

```python
from pathlib import Path
Path("source.bin").write_bytes(bytes(range(256)))
with open("source.bin", "rb") as src:
    with open("copy.bin", "wb") as dst:
        while True:
            block = src.read(64)
            if not block:
                break
            dst.write(block)
print(Path("copy.bin").read_bytes() == bytes(range(256)))
```

output

```
True
```


Slide 77 · Binary File Copying 二进制复制

> 示例构造256种字节并用固定块复制，再比较完整内容。二进制流不做解码和换行转换。嵌套with是已学语法，不需要提前学多个上下文管理器的复杂写法。真实大文件不读全量比较，应按需求使用分块或摘要。

Position and Buffering 文件位置与缓冲

code

```python
with open("data.bin", "w+b") as f:
    f.write(b"0123456789")
    print(f.tell())
    f.seek(2)
    print(f.read(3))
    f.seek(0)
    f.write(b"AB")
    f.flush()
    f.seek(0)
    print(f.read())
```

output

```
10
b'234'
b'AB23456789'
```

78

Slide 78 · Position and Buffering 文件位置与缓冲

> 二进制tell是位置字节数，seek明确定位。文本tell值应当作供seek使用的标记，不随意算术。flush刷新Python缓冲，不承诺设备断电后仍持久。这里以Python实际行为说明读写切换，不照搬C标准流的完全未定义说法。

Regular Expression Syntax 正则语法参考

Syntax	Meaning
[abc] / [a-z] / [^abc]	class / range / complement
* / + / ? / {m,n}	zero+ / one+ / optional / bounded
^ / $	string boundaries; line boundaries with MULTILINE
IGNORECASE / DOTALL	case-insensitive / dot includes newline
\d / \w / \s	Unicode classes for str patterns

79

Slide 79 · Regular Expression Syntax 正则语法参考

> 补集只在字符类起始位置的脱字符有该意义。默认美元还可以匹配字符串末尾换行前，严格整串验证优先fullmatch。DOTALL改变点号，MULTILINE改变锚点，二者不是同一选项。

Match Results and Iteration 匹配结果

code

```python
import re
s = "alice:12 bob:7"
print(re.findall(r"[a-z]+:[0-9]+", s))
print(re.findall(r"([a-z]+):[0-9]+", s))
print(re.findall(r"([a-z]+):([0-9]+)", s))
```

output

```
['alice:12', 'bob:7']
['alice', 'bob']
[('alice', '12'), ('bob', '7')]
```


Slide 80 · Match Results and Iteration 匹配结果

> findall无组返回整体，一个捕获组返回组字符串，多个组返回元组列表。另可用Match.start/end/span定位，finditer按匹配迭代，机制留给L13，不在这里要求解释迭代器协议。模块函数也会缓存最近的模式。

Pattern Tests and Parser Choice 模式测试

- **Specify scope**: common contact forms, not every possible address.
- **Positive cases**: ordinary input and agreed variations.
- **Negative cases**: missing fields, extra suffixes, Unicode surprises.
- **Use a format parser** for CSV, HTML and other structured formats.

Slide 81 · Pattern Tests and Parser Choice 模式测试

> 邮箱练习必须定义接受范围，不能要求学生自己发明完整标准。Python正则含回溯引用等扩展，不直接等同形式语言课中的正则表达式。使用专用解析器是工程选择，本课不靠一条过强的不可表达性断言来证明。

File-Based Membership Benchmark 成员查询实验

- **Generate** database and query files with a recorded seed.
- **Load** the same values into a list, dict and set.
- **Compare** membership results before comparing elapsed time.
- **Scale gradually**: begin with small inputs; separate I/O and lookup time.

Slide 82 · File-Based Membership Benchmark 成员查询实验

> 保留往年上机的核心实验。旧默认一百万数据乘五万列表查询可能很慢，不作为全班必做规模。先小规模验证结果一致，再研究增长；测完整流程和只测查询要明确区分。字典与集合查找为通常平均表现，不作无条件最坏常数保证。

Diagnostic Examples 完整错误示例

Actual Python 3.13 traceback · line-wrapped

```
Traceback (most recent call last):
  File "decode_error.py", line 2, in <module>
    print(data.decode("ascii"))
          ~~~~~~~~~~~^^^^^^^^^
UnicodeDecodeError: 'ascii' codec can't decode byte 0xe4 in position 0:
ordinal not in range(128)
```

保留异常类型、位置与原始输入；不要只截取最后一句。

83

Slide 83 · Diagnostic Examples 完整错误示例

完整输出来自decode_error.py实际运行。显示行折行只是排版，不删异常链和定位信息。缺文件、坏记录、字段值错误的完整案例见讲义。修复应从输入和预期编码出发，而不是把异常统一屏蔽。

References 参考资料

- Python 3.13 · random
- Python 3.13 · re
- Python 3.13 · files
- QED history · Dennis Ritchie
- UTF-8 history · Rob Pike

84

Slide 84 · References 参考资料

官方文档说明Python3.13行为。历史采用亲历资料，不复述未经核实的传闻。进一步阅读可从这些文档与原始历史资料开始。