

上海交通大學
SHANGHAI JIAO TONG UNIVERSITY

Introduction to Computation (CS1602) · Lecture 12

Randomness, Text and Files

Instructor: Tao Huang

Part 5: Robustness and the Real World · Fall 2026

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

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

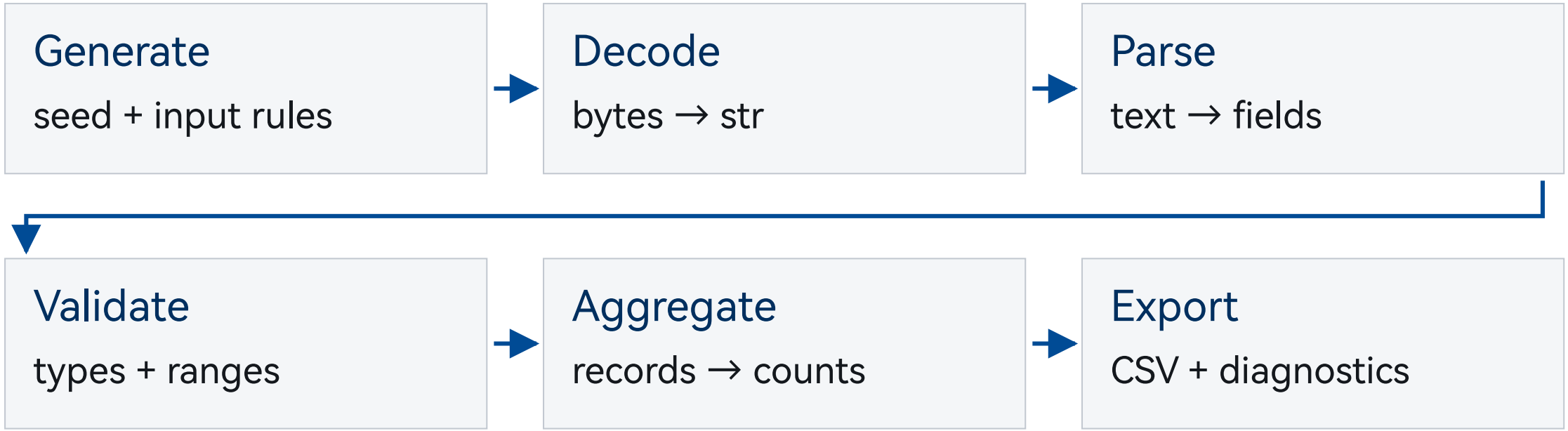
Deliver counts, a CSV file, and useful diagnostics.

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")))
```

The Data Processing Pipeline 处理流程



Random Data for Experiments 随机实验



Simulation: sample possible outcomes.

Testing: exercise different inputs.

Debugging: replay the failing experiment.

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。

Floating-Point Sampling 浮点抽样

Expression	Range / rule
random()	$0.0 \le x < 1.0$
uniform(a, b)	$a + (b - a) \times \text{random}()$
Endpoint b	may occur because of floating-point rounding

Sampling and Replacement 放回与不放回

code

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

output

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

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
```

Weighted Sampling 带权抽样

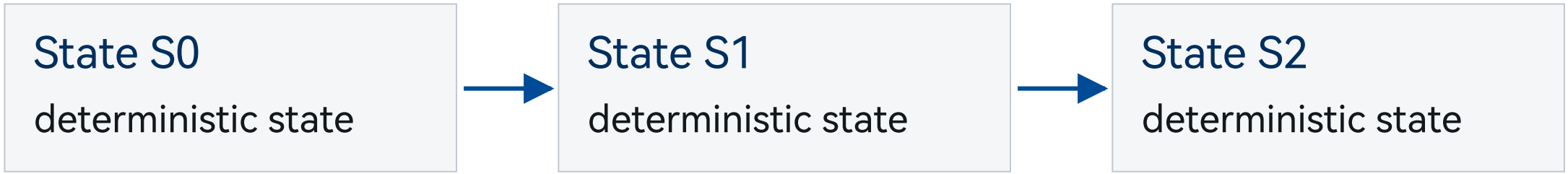
code

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

output

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

Pseudorandom State 伪随机状态



seed → S0

draw 1 → output 1

draw 2 → output 2

Same start + same calls → same sequence

Independent Generators 独立随机对象

code

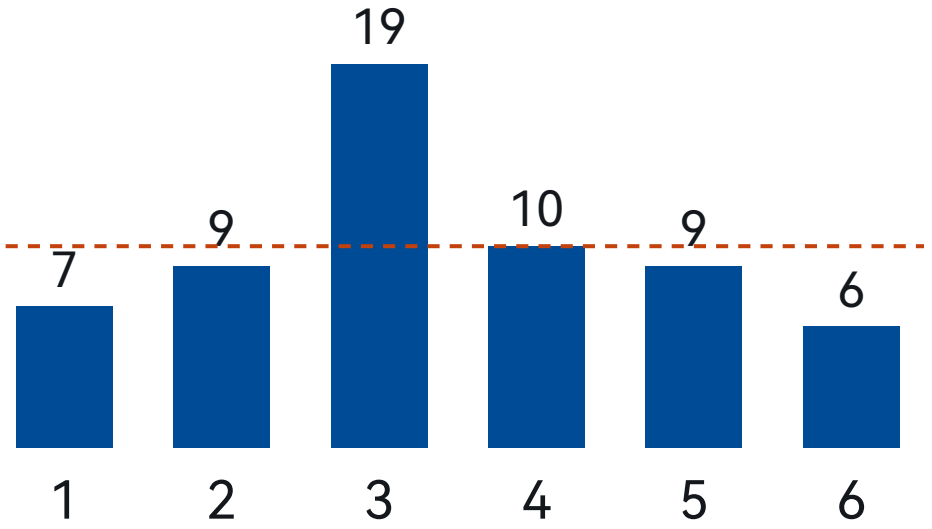
```
1  import random
2  def rolls(seed):
3      rng = random.Random(seed)
4      return [rng.randint(1, 6) for _ in range(5)]
5
6  print(rolls(1602))
7  print(rolls(1602))
8  print(rolls(1602) == rolls(1602))
```

output

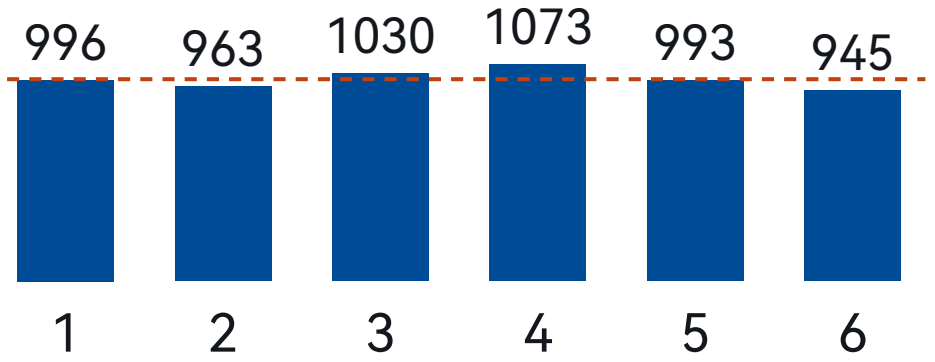
```
[3, 3, 4, 6, 2]
[3, 3, 4, 6, 2]
True
```

Dice Frequencies 骰子频数

n = 60



n = 6000



Orange line: expected frequency 1/6

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

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.

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?

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。

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。

Character Lookup 字符查询

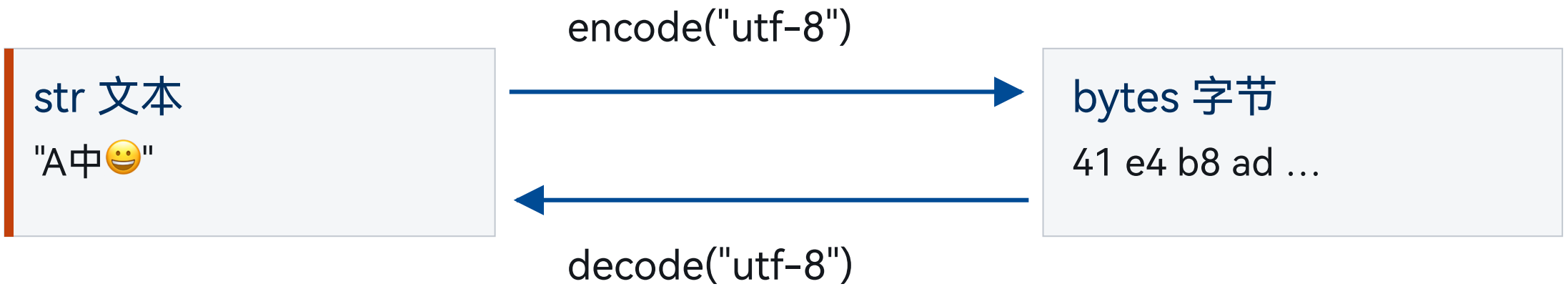
code

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

output

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

Text and Byte Sequences 文本与字节



UTF-8 Byte Lengths 字节长度

A

U+0041

41

中

U+4E2D

e4 b8 ad

😀

U+1F600

f0 9f 98 80

3 code points → 1 + 3 + 4 = 8 bytes

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
```

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。

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

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.

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。

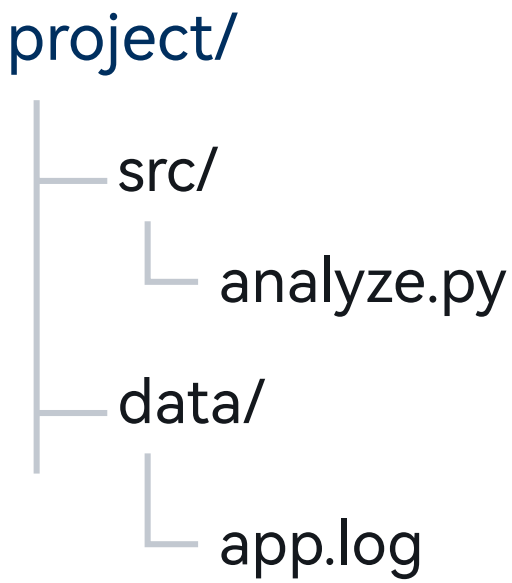
https://taohuang.info/cs1602/zh/quiz/12/

File Processing 文件处理

承上：字符和字节之间需要明确的编码规则。

启下：定位文件、选择模式，并管理读取与关闭。

Paths and the Working Directory 工作目录



cwd: project/

data/app.log → intended file

cwd: project/src/

data/app.log → src/data/app.log

Relative paths start at the working directory. 查不到文件时先确认工作目录。

Paths with pathlib 路径对象

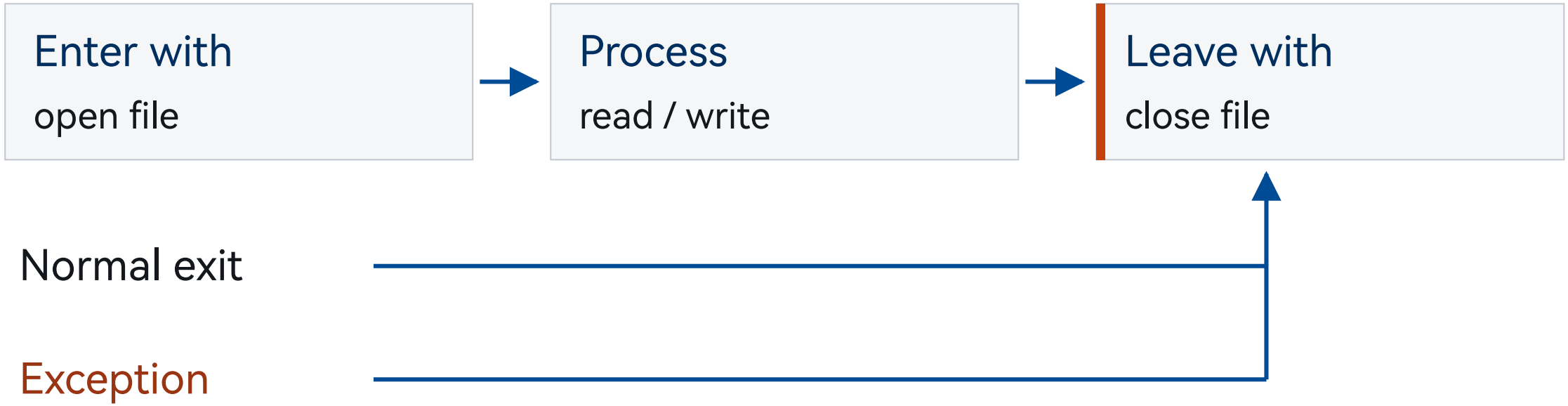
code

```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
```

File Lifetime with with 文件生命周期



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。

File Truncation 文件截断

code

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

output

```
''
```

Writing Text and Newlines 写入与换行

code

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

output

```
'onetwothree'
```

File Reading Methods 文件读取

code

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

output

```
'A中'
'A中\n'
['A中\n', 'last']
```

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'
''
```

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

Whitespace Preservation 空白保留

code

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

output

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

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

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))
```

CSV Quoting 字段中的逗号

code

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

output

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

CSV Reading and Writing 读写闭环

code

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

output

```
True
```

CSV Headers and Field Types 表头与类型

code

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

output

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

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.

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.

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

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']
['.']
```

Character Classes 字符集合

code

```python
import re
for text in ["123", "1 2 3", "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
1 2 3 True False
abc False False
中 False False
```

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

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

Search, Prefix and Full Match 匹配范围

abc12345

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

Missing Matches 未找到匹配

code

```
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'
```

Capturing Groups 捕获分组

code

```python
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'}
```

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)
```

Syntax and Semantic Validation 格式与含义

code

```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
```

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.

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.

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(re.findall(r'"[^"\n]*"', s))
```

output

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

Replacement and Pattern Splitting 替换与切分

code

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

output

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

Quiz 2 · Files and Text Patterns

扫码作答 · 3 题

1 空行、末行与 EOF

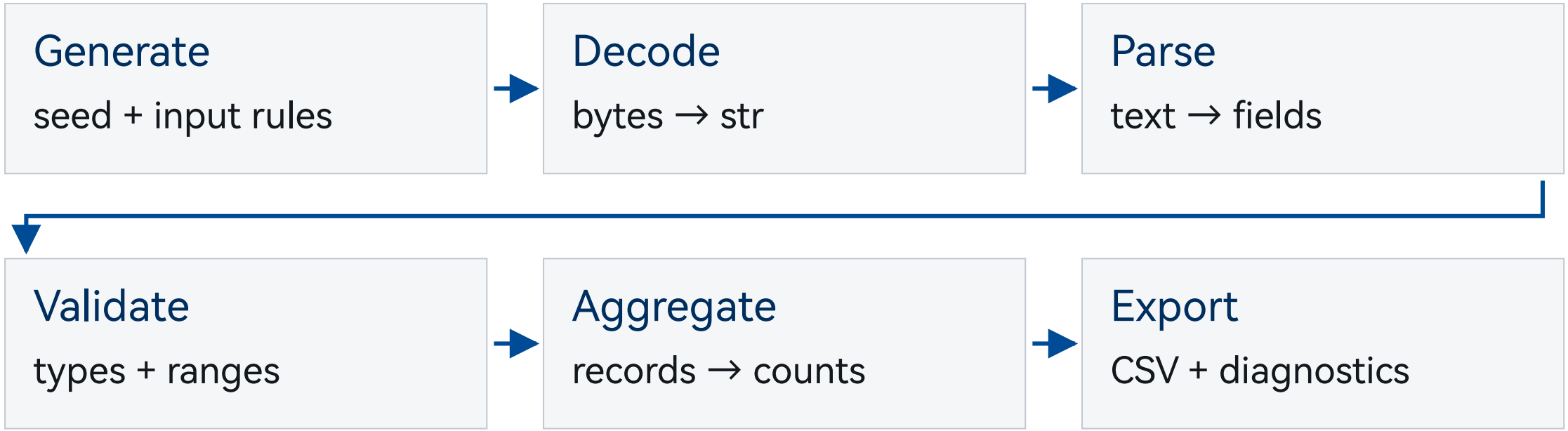
2 局部匹配与整串验证

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/

From Log File to CSV 综合处理



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

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

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.

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

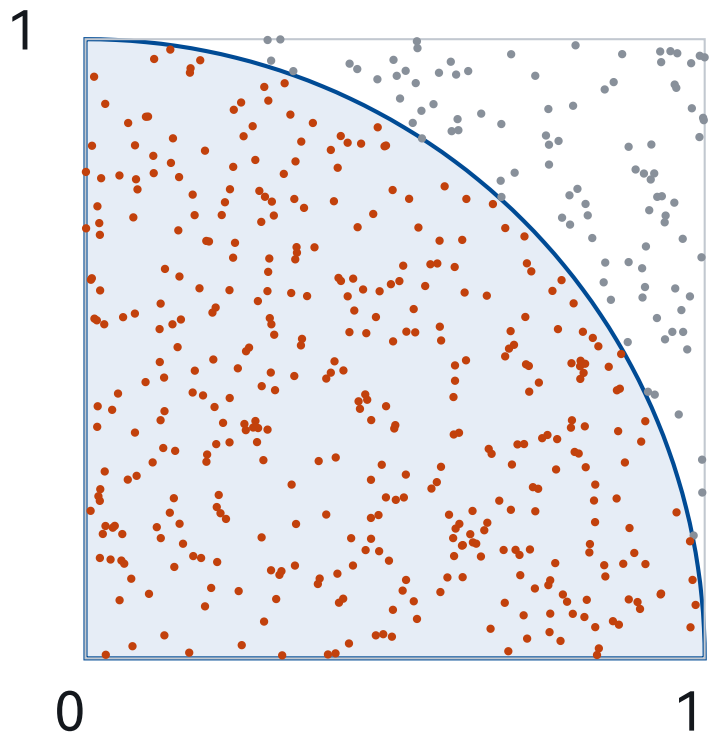
Preview: Iterables and Generators 下讲预告

- **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?

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

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

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.

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
```

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
€
€
```

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
```

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
```

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.

Command-Line Arguments 命令行参数

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

```
1  import sys
2
3  print(sys.argv[0])
4  print(sys.argv[1:])
```

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

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
```

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

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
```

Position and Buffering 文件位置与缓冲

code

```
1  with open("data.bin", "w+b") as f:
2      f.write(b"0123456789")
3      print(f.tell())
4      f.seek(2)
5      print(f.read(3))
6      f.seek(0)
7      f.write(b"AB")
8      f.flush()
9      f.seek(0)
10     print(f.read())
```

output

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

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

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')]
```

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.

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.

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。

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