## | STAT243 Lecture 2.3 Output from Python

# | 1 Writing output to files

- 与读取函数——对应的文本输出接口:
  - open(file\_path, 'w') 写入新文件, 'a' 追加写入。
  - file.write(str\_obj) 写入单个字符串;返回写入的字符数。
  - file.writelines(lines) 批量写入字符串序列(不会自动加换行)。
  - 建议配合 with 使用,确保句柄自动关闭。

```
Python
1 import os
2 file_path = os.path.join('/tmp', 'tmp.txt')
3 with open(file_path, 'w') as file:
       n = file.write('hello\n')
4
       print(n)
5
6 # 输出:
7 # 6
8
   lines = ['alpha\n', 'beta\n']
9
with open(file_path, 'a') as file:
       file.writelines(lines)
11
```

#### • Pandas 输出:

- DataFrame.to\_csv(path, index=False) 输出为 CSV。
- DataFrame.to\_parquet(path) 输出为 Parquet(列式,体积更小,读取更快)。

#### • JSON 输出:

- json.dump(obj, fp) 将 Python dict/list 等序列化为 JSON 内容写入文件。
- 适用于在程序间交换结构化对象(可与 Web API 互操作)。

```
python

import json, os
obj = {'a': 1, 'b': [2, 3]}

path = os.path.join('/tmp', 'obj.json')
with open(path, 'w') as f:
    json.dump(obj, f)
print(os.path.exists(path))
# 输出:
# True
```

### • Pickle (二进制序列化):

• pickle.dump(obj, fp) / pickle.load(fp); 跨平台、可传输复杂 Python 对象。

• 注意安全: 只从可信来源加载 pickle。

```
python

import pickle, os

s = {'k': (1, 2, 3)}

path = os.path.join('/tmp', 'obj.pkl')

with open(path, 'wb') as f:
 pickle.dump(s, f)

print(os.path.getsize(path) > 0)

# 输出:
 # True
```

## 2 Formatting output

- 使用 format 迷你语言 控制对齐、宽度与小数位:
  - {:>10} 右对齐占 10 列; {:.10f} 固定 10 位小数。
  - 函数式接口 format(value, '15.10f') 与等价的 '{:15.10f}'.format(value)。

```
Python
    print('{:>10}'.format(3.5))
1
2 # 输出:
          3.5
3 #
4
5 print('{:.10f}'.format(1/3))
   # 输出:
6
7 # 0.3333333333
8
   print('{:15.10f}'.format(1/3))
9
10
   # 输出:
  # 0.3333333333
11
12
   print(format(1/3, '15.10f'))
13
   # 输出:
14
15 # 0.33333333333
```

• f-strings (Python 3.6+): 在字符串字面量中直接插入表达式。

```
Python

val1, val2 = 1.5, 2.5

print(f"Let's add {val1} and {val2}.")

# 输出:

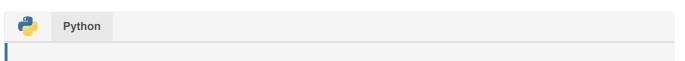
# Let's add 1.5 and 2.5.
```

• 旧式 % 格式化: 保留以兼容历史代码; %s 字符串、%.5f 小数位、%15.7f 宽度与精度。

```
Python

num1 = 1/3
print("Let's add the %s numbers %.5f and %15.7f." % ('floating point', num1, 32+1/7))
# 输出:
# Let's add the floating point numbers 0.33333 and 32.1428571.
```

• 将格式化结果写入文件:



```
import os
file_path = os.path.join('/tmp', 'tmp.txt')
with open(file_path, 'a') as file:
    m = file.write("Let's add the %s numbers %.5f and %15.7f.\n" % ('floating point', num1, 32+1/7))
print(m)
# 输出:
# 55
```

• round() 也可控制显示位数,但更推荐<mark>直接用格式化</mark>确保输出可控。