Python 与大数据分析 -- python基本语法及变量

1. `print()` (内置函数)
    - 用法: `print(*objects, sep=' ', end='\n', file=sys.stdout, flush=False)` - 打印对象到文本流文件，sep是分隔符，end是结尾符。
    - 示例 (来自课件 page 5):

```python
print("Data")
print("whale")
# 输出:
# Data
# whale

print("Data", "whale")
# 输出: Data whale

print("Data", "whale", sep="-*-")
# 输出: Data-*-whale
```

2. `input()` (内置函数)
    - 用法: `input([prompt])` - 从标准输入读取一行文本，并将其作为字符串返回（去掉末尾的换行符）。
    - 示例 (来自课件 page 7, 结合 `.split()`):

```python
a, b = input().split() # 默认以空格分隔
print(f'a = {a}, b = {b}')
# 输入: 1 2
# 输出: a = 1, b = 2
```

3. `.split()` (字符串方法)
    - 用法: `string.split(sep=None, maxsplit=-1)` - 以 `sep` 为分隔符切片 `string`，如果 `sep` 未指定或为 `None`，任何空白字符串都会被视为分隔符。返回一个由切片结果组成的列表。
    - 示例 (来自课件 page 7, 结合 `input()`):

```python
a, b = input().split(",") # 以逗号分隔
# 输入: 1,2
# a 会是 '1', b 会是 '2'
```

4. `import keyword` (导入模块语句)

- 用法: `import module_name` - 导入一个 Python 模块。
- 示例 (来自课件 page 9):

```python
import keyword
```

5. `keyword.kwlist` (模块属性)

- 用法: `keyword.kwlist` - Python `keyword` 模块中的一个列表，包含了 Python 的所有关键字。
- 示例 (来自课件 page 9):

```python
import keyword
kw = keyword.kwlist
print(kw)
# 输出: ['False', 'None', 'True', 'and', 'as', ...]
```

6. `del` (关键字)

- 用法: `del object_name` - 用于删除对象。可以删除单个变量、列表的元素、字典的键值对等。
- 示例 (来自课件 page 10, 删除覆盖了内置函数的变量):

```python
print = 1 # 覆盖了内置的print函数
# print(print) # 这会报错 TypeError
# 解决方法：用del删除自定义变量
del print
print("恢复正常")
```

7. `type()` (内置函数)

- 用法: `type(object)` - 返回对象的类型。
- 示例 (来自课件 page 15):

```python
a = 10
print(type(a))
# 输出: <class 'int'>

b = 1/4
print(type(b))
# 输出: <class 'float'>
```

8. `float()` (内置函数)
    - 用法: `float([x])` - 将字符串或数字转换为浮点数。
    - 示例 (来自课件 page 22):

```python
a = -2023.5
b = int(a) # 转换为整型
print(b, type(b))
# 输出: -2023 <class 'int'>
```

(注意：示例中 float() 的直接使用是在类型转换部分，这里用 int(a) 来说明其相关性)

9. `bool()` (内置函数)
    - 用法: `bool([x])` - 将值转换为布尔值（True 或 False）。
    - 示例 (来自课件 page 22):

```python
a = -2023.5
c = bool(a) # 转换为布尔型
print(c, type(c))
# 输出: True <class 'bool'> (非零数值转为True)
```

10. `int()` (内置函数)
    - 用法: `int([x[, base]])` - 将字符串或数字转换为整数。
    - 示例 (来自课件 page 22):

```python
a = -2023.5
b = int(a) # 转换为整型
print(b, type(b))
# 输出: -2023 <class 'int'>
```

11. `str()` (内置函数)
    - 用法: `str(object='')` - 返回 `object` 的字符串表示形式。

- 示例 (来自课件 page 22):

```python
a = -2023.5
d = str(a) # 转换为字符串型
print(d, type(d))
# 输出: -2023.5 <class 'str'>
```

12. `isinstance()` (内置函数)

- 用法: `isinstance(object, classinfo)` - 如果 `object` 参数是 `classinfo` 参数的实例，或其（直接、间接或虚拟）子类的实例，则返回 `True`。
- 示例 (来自课件 page 19):

```python
print(isinstance("p2s", str))
# 输出: True

import numbers
def isNumber(x):
    return isinstance(x, numbers.Number) # 可以应对任何类型的数字
print(isNumber(1), isNumber(1.1), isNumber(1+2j), isNumber("p2s"))
# 输出: True True True False
```

13. `def` (关键字)

- 用法: `def function_name(parameters): ...` - 用于定义一个函数。
- 示例 (来自课件 page 56, 定义求和函数):

```python
def sum(num1, num2):
    # 两数之和
    return num1 + num2
print(sum(5, 6))
# 输出: 11
```

14. `return` (关键字)

- 用法: `return [expression]` - 退出一个函数，并可选地向调用者传回一个表达式的值。
- 示例 (来自课件 page 56):

```python
def sum(num1, num2):
    return num1 + num2
```

15. `match ... case` (语句, Python 3.10+)

- 用法: 结构化模式匹配，允许你根据值的结构执行不同的代码块。
- 示例 (来自课件 page 32, 语法格式):

```
# 假设 subject 是一个变量
# match subject:
#     case <pattern_1>:
#         <action_1>
#     case <pattern_2>:
#         <action_2>
#     case _: # 通配符，类似 default
#         <action_wildcard>
# 课件中 getGrade(score) 的例子可以用 match...case 重写
def getGrade_match(score):
    match score:
        case s if s >= 90: return "A"
        case s if s >= 80: return "B"
        case s if s >= 70: return "C"
        case s if s >= 60: return "D"
        case _: return "F"
print(getGrade_match(85)) # 输出 B
```

(注意：课件 page 30 的 `getGrade` 是用 `if/elif/else` 实现的，`match...case` 是 Python 3.10 引入的新特性，课件提到了但未给出具体 Python 代码示例，这里补充一个等效示例)

16. `while` (关键字)

- 用法: `while condition: ...` - 只要条件为真，就重复执行循环体内的语句。
- 示例 (来自课件 page 36):

```python
a = 1
while a < 10:
    print(a)
    a += 2
# 输出:
# 1
# 3
# 5
# 7
# 9
```

17. `for ... in ...:` (关键字)

- 用法: `for item in iterable: ...` - 遍历可迭代对象（如列表、字符串、range对象等）中的每个元素。
- 示例 (来自课件 page 40):

```python
for letter in '你好! python':
    print(letter)
# 输出:
# 你
# 好
# !
#
# p
# y
# t
# h
# o
# n
```

18. `range()` (内置函数)

- 用法: `range(stop)` 或 `range(start, stop[, step])` - 生成一个不可变的数字序列，常用于 `for` 循环。
- 示例 (来自课件 page 42):

```python
def sumFromMToN(m, n):
    total = 0
    for x in range(m, n+1): # 注意 n+1 使其包含 n
        total += x
    return total
print(sumFromMToN(5, 10)) # 5+6+7+8+9+10 = 45
# 输出: 45
```

19. `.format()` (字符串方法)

- 用法: `string.format(*args, **kwargs)` - 执行字符串格式化操作。
- 示例 (来自课件 page 45, 打印九九乘法表):

```python
# 打印九九乘法表
for i in range(1, 10):
    for j in range(1, i+1):
        # 在字符串语句中，大括号及其里面的字符 (称作格式化字段) 将会被 format() 中的参数替换。
        print('{}x{}={}\t'.format(j, i, i*j), end='')
    print()
```

20. `break` (关键字)

- 用法: `break` - 立即终止其所在的循环（`for` 或 `while`）。
- 示例 (来自课件 page 49):

```python
for num in range(10, 20): # 迭代 10 到 20 之间的数字
    for i in range(2, num): # 根据因子迭代
        if num % i == 0: # 确定第一个因子
            j = num / i # 计算第二个因子
            print('%d 等于 %d * %d' % (num, i, j))
            break # 跳出当前（内层）循环
    else: # 循环的 else 部分
        print('%d 是一个质数' % num) # 循环中没有 break 时执行
```

21. `continue` (关键字)

- 用法: `continue` - 立即结束当前迭代，并跳到循环的下一次迭代。
- 示例 (来自课件 page 51):

```python
# 统计1到100之间的奇数和，
# 那么也就是说当count是偶数时，需要跳出当次的循环
count = 1
sum_val = 0
while count <= 100:
    if count % 2 == 0: # 判断是否为偶数
        count = count + 1
        continue # 若未偶数跳出当前循环，进入下一次
    sum_val = sum_val + count
    count = count + 1
print(sum_val)
# 输出: 2500
```

22. `pass` (关键字)
    - 用法: `pass` - 空操作语句，用作占位符。
    - 示例 (来自课件 page 52):

```python
for letter in 'Python':
    if letter == 'h':
        print('这是 pass 块')
        pass
    else:
        print('当前字母 :', letter)
# 输出:
# 当前字母 : P
# 当前字母 : y
# 当前字母 : t
# 这是 pass 块
# 当前字母 : o
# 当前字母 : n
```

Python 与大数据分析 -- List，Tuple，Set，和Dict (File 7)

1. `list()` (内置类型/构造函数)
    - 用法: `list([iterable])` - 创建一个列表。如果未提供参数，则创建一个空列表。如果提供了可迭代对象，则从其元素创建列表。
    - 示例 (来自课件 page 3, 创建列表):

```python
list1 = ['physics', 'chemistry', 1997, 2000]
list2 = [1, 2, 3, 4, 5]
list3 = ["a", "b", "c", "d"]
```

- 示例 (来自课件 page 5, 创建空列表的另一种方式):

```
b = list()
print(type(b), len(b), b)
# 输出: <class 'list'> 0 []
```

2. `len()` (内置函数)
    - 用法: `len(s)` - 返回对象的长度（项目数）。
    - 示例 (来自课件 page 4):

```
petInfo = ['cat', 'linus', 14, 15.6, True]
n = len(petInfo)
print(n)
# 输出: 5
```

3. `.append()` (列表方法)
    - 用法: `list.append(x)` - 将 `x` 添加到列表的末尾。
    - 示例 (来自课件 page 14, 添加元素到列表末尾):

```
a = [10, 20, 30]
a.append(10) # Add element to the end
print(a)
# 输出: [10, 20, 30, 10]
```

4. `.insert()` (列表方法)
    - 用法: `list.insert(i, x)` - 在给定位置 `i` 插入元素 `x`。
    - 示例 (来自课件 page 14, 在索引2处插入25):

```
a = [10, 20, 30, 10]
a.insert(2, 25) # Insert 25 into index 2
print(a)
# 输出: [10, 20, 25, 30, 10]
```

5. `.pop()` (列表方法)
    - 用法: `list.pop([i])` - 移除列表中给定位置的元素并返回它。如果未指定索引，则移除并返回最后一个元素。
    - 示例 (来自课件 page 14, 移除最后一个元素):

```
a = [10, 20, 25, 30, 10]
a.pop() # Remove element from the end
print(a)
# 输出: [10, 20, 25, 30]
```

- 示例 (来自课件 page 29, 移除指定索引的元素):

```
item = a.pop(3) # 移除索引为3的元素
print("item =", item)
print("a =", a)
# 假设 a = [2, 3, 4, 5, 6, 7, 8]
# item = 5
# a = [2, 3, 4, 6, 7, 8]
```

6. `.remove()` (列表方法)

- 用法: `list.remove(x)` - 移除列表中第一个值为 `x` 的元素。如果未找到该元素，则引发 `ValueError`。
- 示例 (来自课件 page 14, 移除第一个出现的20):

```
a = [10, 20, 25, 20, 10] # 假设a是这样
a.remove(20) # Remove (first occurrence of) element with value 20
print(a)
# 输出: [10, 25, 20, 10]
```

7. `import copy` (导入模块语句)

- 用法: `import copy` - 导入 `copy` 模块，该模块提供了浅拷贝和深拷贝操作。
- 示例 (来自课件 page 19):

```
import copy
```

8. `copy.copy()` (模块函数 - 浅拷贝)

- 用法: `copy.copy(x)` - 返回 `x` 的浅拷贝。
- 示例 (来自课件 page 19):

```python
import copy
a = [2, 3]
c = copy.copy(a) # Ok (this creates a shallow copy)
a[0] = 42
print("c =", c)
# 输出: c = [2, 3] (c 不会随 a 改变)
```

9. `.count()` (列表方法)
    - 用法: `list.count(x)` - 返回元素 `x` 在列表中出现的次数。
    - 示例 (来自课件 page 24):

```python
a = [2, 3, 5, 2, 6, 2, 2, 7]
print("a.count(2) =", a.count(2))
# 输出: a.count(2) = 4
```

10. `.index()` (列表方法)
    - 用法: `list.index(x[, start[, end]])` - 返回列表中第一个值为 `x` 的元素的索引。如果未找到，则引发 `ValueError`。
    - 示例 (来自课件 page 25):

```python
a = [2, 3, 5, 2, 6, 2, 2, 7]
print("a.index(6) =", a.index(6))
# 输出: a.index(6) = 4
print("a.index(2, 1) =", a.index(2, 1)) # 从索引1开始查找2
# 输出: a.index(2, 1) = 3
```

11. `.extend()` (列表方法)
    - 用法: `list.extend(iterable)` - 通过附加可迭代对象中的元素来扩展列表。
    - 示例 (来自课件 page 26):

```python
a = [2, 3]
a.extend([17, 19])
print(a)
# 输出: [2, 3, 17, 19]
```

12. `.sort()` (列表方法 - 原地排序)

- 用法: `list.sort(key=None, reverse=False)` - 对列表中的项进行原地排序。
- 示例 (来自课件 page 36):

```
a = [7, 2, 5, 3, 5, 11, 7]
a.sort()
print("After a.sort(), a =", a)
# 输出: After a.sort(), a = [2, 3, 5, 5, 7, 7, 11]
```

13. `.reverse()` (列表方法 - 原地反转)
    - 用法: `list.reverse()` - 原地反转列表中的元素。
    - 示例 (来自课件 page 36):

```
a = [2, 3, 5, 7]
a.reverse()
print("Here are the items in reverse:")
for item in a:
    print(item)
# 输出:
# Here are the items in reverse:
# 7
# 5
# 3
# 2
```

14. `tuple()` (内置类型/构造函数)
    - 用法: `tuple([iterable])` - 创建一个元组。如果未提供参数，则创建一个空元组。如果提供了可迭代对象，则从其元素创建元组。
    - 示例 (来自课件 page 38):

```
t = (1, 2, 3)
print(type(t), len(t), t)
# 输出: <class 'tuple'> 3 (1, 2, 3)

a = [1, 2, 3]
t = tuple(a)
print(type(t), len(t), t)
# 输出: <class 'tuple'> 3 (1, 2, 3)
```

15. `set()` (内置类型/构造函数)

- 用法: `set([iterable])` - 创建一个新的集合对象。如果提供了可迭代对象，则从其元素创建集合。
- 示例 (来自课件 page 43):

```
s = set([2,3,5])
print(s)
# 输出: {2, 3, 5} (顺序可能不同)
```

- 示例 (来自课件 page 44, 创建空集合):

```
s = set() # 正确创建空集合的方式
print(s)
# 输出: set()
```

16. `.add()` (集合方法)
    - 用法: `set.add(elem)` - 向集合中添加元素 `elem`。
    - 示例 (来自课件 page 45):

```
s = set((1, 2, 3))
s.add(7)
print(s)
# 输出: {1, 2, 3, 7} (顺序可能不同)
```

17. `.remove()` (集合方法)
    - 用法: `set.remove(elem)` - 从集合中移除元素 `elem`。如果元素不存在，则引发 `KeyError`。
    - 示例 (来自课件 page 45):

```
s = {1, 2, 3, 7}
s.remove(3)
print(s)
# 输出: {1, 2, 7} (顺序可能不同)
```

18. `dict()` (内置类型/构造函数)
    - 用法: `dict(**kwarg)` 或 `dict(mapping, **kwarg)` 或 `dict(iterable, **kwarg)` - 创建一个新的字典。
    - 示例 (来自课件 page 53, 创建空字典):

```python
d = dict()
print(d) # 输出: {}

d = {} # 也可以这样创建空字典
print(d) # 输出: {}
```

- 示例 (来自课件 page 53, 从键值对列表创建字典):

```python
pairs = [("cow", 5), ("dog", 98), ("cat", 1)]
d = dict(pairs)
print(d)
# 输出: {'cow': 5, 'dog': 98, 'cat': 1} (顺序可能不同)
```

19. `.get()` (字典方法)

- 用法: `dict.get(key[, default])` - 返回指定键的值。如果键不存在，则返回 `default` 值（如果已指定），否则返回 `None`。
- 示例 (来自课件 page 54):

```python
d = {"a": 1, "b": 2, "c": 3}
print(d.get("c", 42)) # 如果键存在于字典中，则找到对应的值
# 输出: 3
print(d.get("z", 42)) # 如果键不存在，则返回第二个参数设定的默认值 (在这个例子中是42)
# 输出: 42
```

20. `del` (关键字，用于字典)

- 用法: `del dict[key]` - 删除字典中指定的键值对。
- 示例 (来自课件 page 54):

```python
d = {"a": 1, "b": 2, "c": 3}
del d["c"] # 从字典中删除指定的键值对，如果键不存在则会报错
print(d)
# 输出: {'a': 1, 'b': 2}
```

Python 与大数据分析 --python函数的使用 (File 2, 补充 Lambda, map, filter, sorted)

21. `lambda` (关键字)

- 用法: `lambda arguments: expression` - 创建一个匿名函数。
- 示例 (来自课件 page 40):

```python
# 定义一个简单的Lambda函数，对传入的参数求平方
square = lambda x: x * x
# 调用Lambda函数
result = square(5)
print(result) # 输出: 25
```

22. `map()` (内置函数)

- 用法: `map(function, iterable, ...)` - 将 `function` 应用于 `iterable` 中的每个项目并返回一个迭代器。
- 示例 (来自课件 page 42, 结合 `lambda`):

```python
# 使用Lambda函数和map()函数将列表中的每个元素都平方
numbers = [1, 2, 3, 4, 5]
squared = list(map(lambda x: x * x, numbers))
print(squared) # 输出: [1, 4, 9, 16, 25]
```

23. `filter()` (内置函数)

- 用法: `filter(function, iterable)` - 从 `iterable` 中过滤出 `function` 返回 `True` 的元素，并返回一个迭代器。
- 示例 (来自课件 page 42, 结合 `lambda`):

```python
# 使用Lambda函数和filter()函数筛选出列表中的偶数
numbers = [1, 2, 3, 4, 5]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers))
print(even_numbers) # 输出: [2, 4]
```

24. `sorted()` (内置函数)

- 用法: `sorted(iterable, key=None, reverse=False)` - 从 `iterable` 中的项返回一个新的已排序列表。
- 示例 (来自课件 page 43, 结合 `lambda` 作为 `key`):

```python
# 使用Lambda函数定义一个自定义的排序规则
students = [
    {'name': 'Tiyong', 'grade': 90},
    {'name': 'Bob', 'grade': 85},
    {'name': 'Toy', 'grade': 95}
]
# 按照学生的成绩进行排序
sorted_students = sorted(students, key=lambda x: x['grade'], reverse=True)
print(sorted_students)
# 输出: [{'name': 'Toy', 'grade': 95}, {'name': 'Tiyong', 'grade': 90}, {'name': 'Bob', 'grade': 85}]
```

Python 与大数据分析 -- NumPy介绍、安装及其操作 (File 13)

1. `import numpy as np` (导入模块语句)
    - 用法: 导入 NumPy 库并使用别名 `np`。
    - 示例 (来自课件 page 5):

```python
import numpy as np
```

2. `np.array()`
    - 用法: `np.array(object, dtype=None, ...)` - 创建一个 NumPy 数组 (ndarray)。
    - 示例 (来自课件 page 10, 创建一维数组):

```python
a = np.array([1, 2, 3, 4, 5]) # 使用列表构建一维数组
print(a)
# 输出: [1 2 3 4 5]
```

    - 示例 (来自课件 page 10, 创建多维数组):

```python
b = np.array([[1, 2, 3], [4, 5, 6]])
print(b)
# 输出:
# [[1 2 3]
#  [4 5 6]]
```

3. `.dtype` (ndarray 属性)

- 用法: `array.dtype` - 返回数组中元素的数据类型。
- 示例 (来自课件 page 8):

```
x = np.array([1, 2])
print(x.dtype) # 输出: int32 (或 int64, 取决于系统)
x = np.array([1.0, 2.0])
print(x.dtype) # 输出: float64
```

4. `.astype()` (ndarray 方法)

- 用法: `array.astype(dtype, casting='unsafe', copy=True)` - 创建一个具有指定数据类型的新数组，并复制原数组的数据。
- 示例 (来自课件 page 9):

```
arr = np.array([1, 2, 3, 4, 5])
print(arr.dtype) # 输出: int32 (或 int64)
float_arr = arr.astype(np.float64)
print(float_arr.dtype) # 输出: float64
```

5. `.shape` (ndarray 属性)

- 用法: `array.shape` - 返回一个表示数组维度的元组。
- 示例 (来自课件 page 11, 一维数组):

```
a = np.array([0, 1, 2, 3])
print(a.shape)
# 输出: (4,)
```

- 示例 (来自课件 page 12, 二维数组):

```
a = np.array([[0, 1, 2, 3], [10, 11, 12, 13]])
print(a.shape)
# 输出: (2, 4)
```

6. `.size` (ndarray 属性)

- 用法: `array.size` - 返回数组中元素的总数。
- 示例 (来自课件 page 11):

```python
a = np.array([0, 1, 2, 3])
print(a.size)
# 输出: 4
```

7. `np.arange()`
    - 用法: `np.arange([start,] stop[, step,], dtype=None)` - 在给定间隔内返回均匀间隔的值。
    - 示例 (来自课件 page 13):

```python
x = np.arange(8) # 创建区间数组
print(x)
# 输出: [0 1 2 3 4 5 6 7]

x = np.arange(1, 10, 2) # 设定步长
print(x)
# 输出: [1 3 5 7 9]
```

8. `np.ones()`
    - 用法: `np.ones(shape, dtype=None, order='C')` - 返回给定形状和类型的新数组，用 1 填充。
    - 示例 (来自课件 page 14):

```python
print(np.ones((2, 4)))
# 输出:
# [[1. 1. 1. 1.]
#  [1. 1. 1. 1.]]
```

9. `np.zeros()`
    - 用法: `np.zeros(shape, dtype=float, order='C')` - 返回给定形状和类型的新数组，用 0 填充。
    - 示例 (来自课件 page 14):

```python
print(np.zeros((3, 2)))
# 输出:
# [[0. 0.]
#  [0. 0.]
#  [0. 0.]]
```

10. `np.eye()`

- 用法: `np.eye(N, M=None, k=0, dtype=float, order='C')` - 返回一个二维数组，对角线为 1，其他地方为 0。
- 示例 (来自课件 page 14):

```python
print(np.eye(4))
# 输出:
# [[1. 0. 0. 0.]
#  [0. 1. 0. 0.]
#  [0. 0. 1. 0.]
#  [0. 0. 0. 1.]]
```

11. `.reshape()` (ndarray 方法) / `np.reshape()`
    - 用法: `array.reshape(newshape, order='C')` 或 `np.reshape(array, newshape, order='C')` - 在不更改其数据的情况下为数组赋予新的形状。
    - 示例 (来自课件 page 15):

```python
e = np.array([[1, 2], [3, 4], [5, 6]])
print("原数组:", e)
# 原数组:
# [[1 2]
#  [3 4]
#  [5 6]]
e_reshaped = e.reshape(2, 3)
print("新数组:", e_reshaped)
# 新数组:
# [[1 2 3]
#  [4 5 6]]
```

12. `.resize()` (ndarray 方法)
    - 用法: `array.resize(new_shape, refcheck=True)` - 原地更改数组的形状和大小。
    - 示例 (来自课件 page 17):

```python
a = np.array([[1, 2], [3, 4], [5, 6]])
a.resize(2, 3) # 原地修改 a
print(a)
# 输出:
# [[1 2 3]
#  [4 5 6]]
a.resize(2, 2, refcheck=False) # 如果新大小不同，需要 refcheck=False
print(a) # 会截断或补零
# 输出 (截断):
# [[1 2]
#  [3 4]]
```

13. `np.resize()`

- 用法: `np.resize(a, new_shape)` - 返回具有指定形状的新数组。如果新数组大于原始数组，则新数组将填充 `a` 的重复副本。
- 示例 (来自课件 page 18):

```python
a = np.array([[1, 2], [3, 4], [5, 6]])
b = np.resize(a, (2, 2))
print(b)
# 输出:
# [[1 2]
#  [3 4]]
b = np.resize(a, (2, 5))
print(b) # 会用 a 的重复副本填充
# 输出:
# [[1 2 3 4 5]
#  [6 1 2 3 4]]
```

14. `slice()` (内置函数)

- 用法: `slice(stop)` 或 `slice(start, stop[, step])` - 返回一个表示由 `range(start, stop, step)` 指定的索引集的 slice 对象。
- 示例 (来自课件 page 20):

```python
s = slice(2, 9, 3) # 从索引2开始到索引9停止，步幅为3
a = np.arange(0, 90, 10)
print(a[s])
# 输出: [20 50 80]
```

15. `.copy()` (ndarray 方法)

- 用法: `array.copy(order='C')` - 返回数组的副本。
- 示例 (来自课件 page 22):

```python
a = np.arange(0, 90, 10)
a_copy = a[5:8].copy() # 创建切片的副本
a_copy = a_copy + 10
print(a_copy) # 输出: [60 70 80]
print(a) # 原数组 a 不受影响
# 输出: [ 0 10 20 30 40 50 60 70 80]
```

16. `.T` (ndarray 属性)

- 用法: `array.T` - 数组的转置。
- 示例 (来自课件 page 38):

```python
arr = np.arange(15).reshape((3, 5))
print("arr:\n", arr)
print("arr.T:\n", arr.T)
# arr:
#  [[ 0  1  2  3  4]
#   [ 5  6  7  8  9]
#   [10 11 12 13 14]]
# arr.T:
#  [[ 0  5 10]
#   [ 1  6 11]
#   [ 2  7 12]
#   [ 3  8 13]
#   [ 4  9 14]]
```

17. `np.where()`

- 用法: `np.where(condition[, x, y])` - 根据条件返回从 `x` (如果为 True) 或 `y` (如果为 False) 中选择的元素。如果只给出条件，则返回 `condition.nonzero()`。
- 示例 (来自课件 page 37):

```python
a = np.arange(0, 100, 10)
a[5] = np.nan # 假设我们手动设置一个 NaN 来模拟缺失值
condition = a < 50
print(np.where(condition)) # 返回满足条件的元素的索引
# 输出: (array([0, 1, 2, 3, 4]),)
b = np.where(condition, a, 0) # 如果条件为真，取a的值，否则取0
print(b)
# 输出: [ 0. 10. 20. 30. 40.  0.  0.  0.  0.  0.] (假设a[5]不是nan)
```

- 示例 (来自课件 page 79, 查找 NaN 位置):

```python
sepalLength = np.array([5.1, 4.9, np.nan, 4.6, 5.0]) # 示例数据
x = np.isnan(sepalLength)
print('缺失位置：', np.where(x), '\t', x)
# 输出: 缺失位置：  (array([2]),)      [False False True False False]
```

18. `np.dot()`

- 用法: `np.dot(a, b, out=None)` - 两个数组的点积。
- 示例 (来自课件 page 38, 矩阵内积):

```python
arr = np.arange(15).reshape((3, 5))
print(np.dot(arr.T, arr))
```

- 示例 (来自课件 page 64, 向量点积):

```python
a_vec = np.array([1, 2, 3, 4])
b_vec = np.array([1, 2, 0, 3]) # 假设
print(np.dot(a_vec, b_vec)) # 1*1 + 2*2 + 3*0 + 4*3 = 1+4+0+12 = 17
```

19. `np.add()` (ufunc)

- 用法: `np.add(x1, x2, /, out=None, *, where=True, ...)` - 逐元素相加。
- 示例 (来自课件 page 39):

```python
a = np.array([1, 2, 3, 4])
b = np.array([2, 3, 4, 5])
print(a + b)          # 输出: [3 5 7 9]
print(np.add(a, b))   # 输出: [3 5 7 9]
```

20. `np.subtract()` (ufunc)

- 用法: `np.subtract(x1, x2, /, out=None, *, where=True, ...)` - 逐元素相减。
- 示例 (来自课件 page 39):

```
a = np.array([1, 2, 3, 4])
b = np.array([2, 3, 4, 5])
print(a - b)              # 输出: [-1 -1 -1 -1]
print(np.subtract(a, b))  # 输出: [-1 -1 -1 -1]
```

21. `np.equal()` (ufunc)

- 用法: `np.equal(x1, x2, /, out=None, *, where=True, ...)` - 逐元素比较是否相等。
- 示例 (来自课件 page 40):

```
a = np.array([[1, 2, 3, 4], [2, 3, 4, 5]])
b = np.array([[1, 2, 5, 4], [1, 3, 4, 5]])
print(a == b)
# 输出:
# [[ True  True False  True]
#  [False  True  True  True]]
print(np.equal(a,b)) # 同上
```

22. `np.char.add()`

- 用法: `np.char.add(x1, x2)` - 逐元素字符串连接。
- 示例 (来自课件 page 44):

```
s1 = ['I', 'finance', 'Python']
s2 = [' love', ' and', '!']
print(np.char.add(s1, s2))
# 输出: ['I love' 'finance and' 'Python!']
```

23. `np.char.multiply()`

- 用法: `np.char.multiply(a, i)` - 返回 `a` 中的字符串重复 `i` 次。
- 示例 (来自课件 page 45):

```
s = ['hello the world']
print(np.char.multiply(s, 2))
# 输出: array(['hello the worldhello the world'],
dtype='<U30')
```

24. `np.char.split()`

- 用法: `np.char.split(a, sep=None, maxsplit=None)` - 对于 `a` 中的每个元素，返回一个由字符串中单词组成的列表，使用 `sep` 作为分隔符字符串。
- 示例 (来自课件 page 45):

```
s = ['hello,the,world']
print(np.char.split(s, sep=','))
# 输出: array([list(['hello', 'the', 'world'])],
dtype=object)
```

25. `np.char.lower()`

- 用法: `np.char.lower(a)` - 返回数组元素的副本，其中所有基于大小写的字符都已转换为小写。
- 示例 (来自课件 page 46):

```
s1 = 'NumPy'
print(np.char.lower(s1))
# 输出: array('numpy', dtype='<U5')
```

26. `np.char.upper()`

- 用法: `np.char.upper(a)` - 返回数组元素的副本，其中所有基于大小写的字符都已转换为大写。
- 示例 (来自课件 page 48, 内部使用):

```
list1 = np.array(['Li', 'Zeng', 'jiao', 'chen',
'Meng'])
# ...
res.append((np.char.upper(list1[i])) in
np.char.upper(list2))
```

27. `np.sort()`

- 用法: `np.sort(a, axis=-1, kind=None, order=None)` - 返回输入数组的排序副本。

- 示例 (来自课件 page 49):

```
a = np.array([[4,3,1],[9,8,5]])
print(np.sort(a, axis=0)) # 按列排序
# 输出:
# [[4 3 1]
#  [9 8 5]]
print(np.sort(a, axis=1)) # 按行排序
# 输出:
# [[1 3 4]
#  [5 8 9]]
```

28. `np.argsort()`

- 用法: `np.argsort(a, axis=-1, kind=None, order=None)` - 返回沿指定轴对数组进行排序的索引。
- 示例 (来自课件 page 50):

```
A = np.array([[15, 18, 14, 14, 10], [19, 20, 15, 15, 19]]) # 示例数据
A2 = np.argsort(A, axis=0)
print(A2)
# 输出 (表示每列排序后的原始索引):
# [[0 0 1 0 0]
#  [1 1 0 1 1]]
```

29. `.view()` (ndarray 方法)

- 用法: `array.view(dtype=None, type=None)` - 返回具有相同数据的新数组对象，但具有不同的数据类型或形状。这是一个视图，修改视图会影响原始数组。
- 示例 (来自课件 page 52):

```
a = np.arange(12).reshape(3,4)
b = a.view()
b[0,0] = 5
print("a:\n", a) # a 的第一个元素也会变成 5
# a:
#  [[ 5  1  2  3]
#   [ 4  5  6  7]
#   [ 8  9 10 11]]
```

30. `np.append()` (与列表的 `append` 不同)

- 用法: `np.append(arr, values, axis=None)` - 将值附加到数组的末尾。如果 `axis` 未指定，则在附加之前将 `arr` 和 `values` 展平。
- 示例 (来自课件 page 55):

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],
[11,12,13,14,15]])
b = np.zeros((3,2))
c = np.ones((5))
print(np.append(a, b, axis=1)) # 按列合并 (水平方向)
# print(np.append(a, c, axis=0)) # 这会报错，因为 c
不是二维的
# 需要将 c reshape
c_reshaped = c.reshape(1,5)
print(np.append(a,c_reshaped, axis=0)) # 按行合并
(垂直方向)
```

- 示例 (来自课件 page 75, 连接数据和标签):

```
# data 和 p 都是 numpy 数组
# iris_data = np.append(data, p, axis=1)
```

31. `np.concatenate()`

- 用法: `np.concatenate((a1, a2, ...), axis=0, out=None, dtype=None, casting="same_kind")` - 沿现有轴连接一系列数组。
- 示例 (来自课件 page 57):

```
A = np.arange(0,8).reshape(2,4)
B = np.arange(4).reshape(1,4)
print(np.concatenate((A,B), axis=0)) # 按行连接
# 输出:
# [[0 1 2 3]
#  [4 5 6 7]
#  [0 1 2 3]]
C = np.arange(2).reshape(2,1)
print(np.concatenate((A,C), axis=1)) # 按列连接
# 输出:
# [[0 1 2 3 0]
#  [4 5 6 7 1]]
```

32. `np.hstack()`

- 用法: `np.hstack(tup)` - 按水平顺序（列式）堆叠数组。

- 示例 (来自课件 page 59):

```
a = np.array([[1,2,3],[4,5,6],[9,10,11]])
c = np.array([[1,2],[3,4],[5,6]])
print(np.hstack((a, c)))
# 输出:
# [[ 1  2  3  1  2]
#  [ 4  5  6  3  4]
#  [ 9 10 11  5  6]]
```

33. `np.column_stack()`

- 用法: `np.column_stack(tup)` - 将一维数组作为列堆叠到二维数组中。
- 示例 (来自课件 page 60):

```
a = np.array([[1,2,3],[5,6,7],[9,10,11]])
c = np.array([4, (3), 5]) # 一维数组
print(np.column_stack((a, c)))
# 输出:
# [[ 1  2  3  4]
#  [ 5  6  7  3]
#  [ 9 10 11  5]]
```

34. `np.c_[]` (NumPy 特殊索引对象)

- 用法: `np.c_[array1, array2, ...]` - 将切片对象转换为沿第二个轴的串联。
- 示例 (来自课件 page 61, 与 `np.hstack` 和 `np.column_stack` 一起列出):

```
a1 = np.arange(0,8).reshape(2,4)
a3 = np.arange(2).reshape(2,1)
print(np.c_[a1, a3])
# 输出:
# [[0 1 2 3 0]
#  [4 5 6 7 1]]
```

35. `np.vstack()`

- 用法: `np.vstack(tup)` - 按垂直顺序（行式）堆叠数组。
- 示例 (来自课件 page 59):

```
a = np.array([[1,2,3],[5,6,7],[9,10,11]])
b = np.array([[1,2,3],[5,6,7]])
print(np.vstack((a, b)))
# 输出:
# [[ 1  2  3]
#  [ 5  6  7]
#  [ 9 10 11]
#  [ 1  2  3]
#  [ 5  6  7]]
```

36. `np.row_stack()`

- 用法: `np.row_stack(tup)` - 按行堆叠数组（与 `vstack` 类似）。
- 示例 (来自课件 page 61, 与 `np.vstack` 一起列出):

```
a1 = np.arange(0,8).reshape(2,4)
a2 = np.arange(4).reshape(1,4)
print(np.row_stack((a1, a2)))
# 输出:
# [[0 1 2 3]
#  [4 5 6 7]
#  [0 1 2 3]]
```

37. `np.r_[]` (NumPy 特殊索引对象)

- 用法: `np.r_[array1, array2, ...]` - 将切片对象转换为沿第一个轴的串联。
- 示例 (来自课件 page 61, 与 `np.vstack` 和 `np.row_stack` 一起列出):

```
a1 = np.arange(0,8).reshape(2,4)
a2 = np.arange(4).reshape(1,4)
print(np.r_[a1, a2])
# 输出:
# [[0 1 2 3]
#  [4 5 6 7]
#  [0 1 2 3]]
```

38. `np.delete()`

- 用法: `np.delete(arr, obj, axis=None)` - 返回一个新数组，其中沿轴删除了子数组。
- 示例 (来自课件 page 62, 删除列):

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],
[11,12,13,14,15]])
print(np.delete(a, [1, 3], axis=1)) # 删除第1列和第3列 (0-indexed)
# 输出:
# [[ 1  3  5]
#  [ 6  8 10]
#  [11 13 15]]
```

- 示例 (来自课件 page 62, 删除行):

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],
[11,12,13,14,15]])
print(np.delete(a, 1, axis=0)) # 删除第1行 (0-indexed)
# 输出:
# [[ 1  2  3  4  5]
#  [11 12 13 14 15]]
```

39. `np.s_[]` (NumPy 特殊索引对象)

- 用法: `np.s_[obj]` - 构建切片对象的更简洁方法。
- 示例 (来自课件 page 62, 用于连续列删除):

```
a = np.array([[1,2,3,4,5],[6,7,8,9,10],
[11,12,13,14,15]])
print(np.delete(a, np.s_[1:3], axis=1)) # 删除第1列到第2列 (不包括第3列)
# 输出:
# [[ 1  4  5]
#  [ 6  9 10]
#  [11 14 15]]
```

40. `np.cross()`

- 用法: `np.cross(a, b, axisa=-1, axisb=-1, axisc=-1, axis=None)` - 返回两个（向量）数组的叉积。
- 示例 (来自课件 page 64):

```
a_vec = np.array([2, 0, 0])
b_vec = np.array([0, 3, 0])
print(np.cross(a_vec, b_vec))
# 输出: [0 0 6]
```

41. `np.matmul()`

- 用法: `np.matmul(x1, x2, /, out=None, *, casting='same_kind', order='K', dtype=None, subok=True)` - 两个数组的矩阵乘积。
- 示例 (来自课件 page 66):

```
m = np.arange(1,7).reshape(2,3)
n = np.arange(1,13).reshape(3,4)
print(np.matmul(m,n))
# 输出:
# [[ 38  44  50  56]
#  [ 83  98 113 128]]
```

42. `np.vdot()`

- 用法: `np.vdot(a, b)` - 返回两个向量的点积。如果第一个参数是复数，则其共轭用于计算。
- 示例 (来自课件 page 67):

```
a1 = np.arange(1,10).reshape(3,3)
n1 = np.arange(1,10).reshape(3,3)
print(np.vdot(a1,n1)) # 1*1 + 2*2 + ... + 9*9
# 输出: 285
```

43. `np.linalg.det()`

- 用法: `np.linalg.det(a)` - 计算数组的行列式。
- 示例 (来自课件 page 68):

```
a = np.array([[3,2,5],[2,1,4],[4,2,6]])
print(np.linalg.det(a))
# 输出: 2.0
```

44. `np.linalg.inv()`

- 用法: `np.linalg.inv(a)` - 计算（乘法）矩阵的逆。
- 示例 (来自课件 page 68):

```
a = np.array([[3,2,5],[2,1,4],[4,2,6]])
a_inv = np.linalg.inv(a)
print(a_inv)
# 输出:
# [[-1.   1.   1.5]
#  [ 2.  -1.  -1. ]
#  [ 0.   1.  -0.5]]
```

45. `np.linalg.eig()`

- 用法: `np.linalg.eig(a)` - 计算方阵的特征值和右特征向量。
- 示例 (来自课件 page 69):

```
m = np.arange(1,10).reshape(3,3)
eigenvalues, eigenvectors = np.linalg.eig(m)
print("eigenvalues:\n", eigenvalues)
print("eigenvectors:\n", eigenvectors)
```

46. `np.linalg.solve()`

- 用法: `np.linalg.solve(a, b)` - 求解线性矩阵方程或线性标量方程组。
- 示例 (来自课件 page 70):

```
a = np.array([[1,2,3],[4,5,6],[7,8,9]])
b = np.array([1,2,3])
x = np.linalg.solve(a,b) # 求解 Ax = b
print(x)
# 注意：示例中的矩阵 a 是奇异的，所以这个例子会报错。
# 一个可解的例子：
# a = np.array([[3,1], [1,2]])
# b = np.array([9,8])
# x = np.linalg.solve(a,b)
# print(x) # 输出: [2. 3.]
```

47. `np.sum()` (ufunc)

- 用法: `np.sum(a, axis=None, dtype=None, out=None, keepdims=<no value>, initial=<no value>, where=<no value>)` - 对给定轴上的数组元素求和。
- 示例 (来自课件 page 71, MSE 公式):

```
# error = (1 / n) * np.sum ( np.square
(predictions - labels) )
```

48. `np.square()` (ufunc)

- 用法: `np.square(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True)` - 返回输入的逐元素平方。
- 示例 (来自课件 page 71, MSE 公式):

```
# error = (1 / n) * np.sum ( np.square (predictions - labels) )
```

49. `np.mean()`

- 用法: `np.mean(a, axis=None, dtype=None, out=None, keepdims=<no value>, *, where=<no value>)` - 计算沿指定轴的算术平均值。
- 示例 (来自课件 page 76):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
print('萼片长度平均值：', np.mean(sepalLength))
```

50. `np.average()`

- 用法: `np.average(a, axis=None, weights=None, returned=False)` - 计算沿指定轴的加权平均值。
- 示例 (来自课件 page 76):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
print('萼片长度平均值：', np.average(sepalLength))
```

51. `np.median()`

- 用法: `np.median(a, axis=None, out=None, overwrite_input=False, keepdims=False)` - 计算沿指定轴的中值。
- 示例 (来自课件 page 76):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
print('萼片长度中位数：', np.median(sepalLength))
```

52. `np.percentile()`

- 用法: `np.percentile(a, q, axis=None, out=None, overwrite_input=False, method='linear', keepdims=False, *, interpolation=None)` - 计算数据沿指定轴的第 q 个百分位数。
- 示例 (来自课件 page 76):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
print('萼片长度中位数：', np.percentile(sepalLength, 50)) # 50百分位数即中位数
```

- 示例 (来自课件 page 78):

```
x = np.percentile(sepalLength, [5, 95])
print('鸢尾萼片长度第5和第95分位数据：', x)
```

53. `np.std()`

- 用法: `np.std(a, axis=None, dtype=None, out=None, ddof=0, keepdims=<no value>, *, where=<no value>)` - 计算沿指定轴的标准差。
- 示例 (来自课件 page 76):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
print('萼片长度标准差：', np.std(sepalLength))
```

54. `np.amax()`

- 用法: `np.amax(a, axis=None, out=None, keepdims=<no value>, initial=<no value>, where=<no value>)` - 返回数组的最大值或沿轴的最大值。
- 示例 (来自课件 page 77):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
aMax = np.amax(sepalLength)
print(aMax)
```

55. `np.amin()`

- 用法: `np.amin(a, axis=None, out=None, keepdims=<no value>, initial=<no value>, where=<no value>)` - 返回数组的最小值或沿轴的最小值。

- 示例 (来自课件 page 77):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
aMin = np.amin(sepalLength)
print(aMin)
```

56. `np.ptp()`

- 用法: `np.ptp(a, axis=None, out=None, keepdims=<no value>)` - 沿轴的值范围（最大值 - 最小值）。
- 示例 (来自课件 page 77):

```
sepalLength = np.array([5.1, 4.9, 4.7, 4.6, 5.0])
# 示例数据
# x = (sepalLength - np.amin(sepalLength)) /
np.ptp(sepalLength)
```

57. `np.isnan()` (ufunc)

- 用法: `np.isnan(x, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True)` - 对数组元素进行逐个测试，看它们是否为 NaN，并以布尔数组形式返回结果。
- 示例 (来自课件 page 79):

```
sepalLength = np.array([5.1, np.nan, 4.7]) # 示例数据
x = np.isnan(sepalLength)
print(x) # 输出: [False  True False]
```

58. `np.logical_and()` (ufunc)

- 用法: `np.logical_and(x1, x2, /, out=None, *, where=True, casting='same_kind', order='K', dtype=None, subok=True)` - 计算 x1 AND x2 的逐元素真值。
- 示例 (来自课件 page 80):

```
petallength = np.array([1.4, 1.6, 1.3]) # 示例数据
sepalLength = np.array([5.1, 4.9, 4.0]) # 示例数据
index = np.where(np.logical_and(petallength > 1.5,
sepalLength < 5.0))
```

59. `np.any()`

- 用法: `np.any(a, axis=None, out=None, keepdims=<no value>, *, where=<no value>)` - 测试沿给定轴的任何数组元素是否计算为 True。
- 示例 (来自课件 page 81):

```
x = np.array([False, True, False]) # 示例布尔数组
print('是否有缺失值：', np.any(x)) # 输出：是否有缺失值： True
```

60. `[:,np.newaxis]` (NumPy 索引技巧)

- 用法: 在数组切片中使用 `np.newaxis` 可以增加一个新的维度。
- 示例 (来自课件 page 82):

```
volume = np.array([10, 20, 30]) # 一维数组
volume = volume[:, np.newaxis] # 转换为列向量 (3,1)
print(volume.shape) # 输出: (3, 1)
```

Python 与大数据分析 -- pandas的使用 (File 11)

1. `import pandas as pd` (导入模块语句)

- 用法: 导入 Pandas 库并使用别名 `pd`。
- 示例 (来自课件 page 3):

```
import pandas as pd
```

2. `pd.Series()`

- 用法: `pd.Series(data=None, index=None, dtype=None, name=None, copy=False, fastpath=False)` - 创建一个一维带标签的数组（Series）。
- 示例 (来自课件 page 6, 创建空 Series):

```
s = pd.Series()
print(s)
```

- 示例 (来自课件 page 7, 从列表创建 Series):

```
s1 = pd.Series([47, 66, 48, 77, 16, 91])
print(s1)
```

- 示例 (来自课件 page 9, 创建带自定义索引的 Series):

```
s2 = pd.Series([47, 66, 48, 77, 16, 91], index=['a','b','c','d','e','f'])
print(s2)
```

3. `.values` (Series 属性)

- 用法: `Series.values` - 返回 Series 的 NumPy 表示形式（即其数据）。
- 示例 (来自课件 page 8):

```
s1 = pd.Series([47, 66, 48, 77, 16, 91])
print(s1.values)
# 输出: [47 66 48 77 16 91]
```

4. `.index` (Series/DataFrame 属性)

- 用法: `Series.index` 或 `DataFrame.index` - 返回 Series 或 DataFrame 的索引。
- 示例 (来自课件 page 8, Series 索引):

```
s1 = pd.Series([47, 66, 48, 77, 16, 91])
print(s1.index)
# 输出: RangeIndex(start=0, stop=6, step=1)
```

- 示例 (来自课件 page 16, 修改 Series 索引):

```
s4 = pd.Series({'ID': np.nan, 'Name': 'Zhang San', 'Height': 178, 'Weight': 66})
s4.index = ['Number', 'Name', 'Weight', 'Height']
# 直接赋值修改索引
print(s4)
```

5. `pd.isnull()`

- 用法: `pd.isnull(obj)` - 检测类似数组的对象中的缺失值（NaN）。
- 示例 (来自课件 page 17):

```python
s4 = pd.Series({'Number': np.nan, 'Name': 'Zhang San', 'Weight': 178, 'Height': 66})
print(pd.isnull(s4))
# 输出:
# Number     True
# Name      False
# Weight    False
# Height    False
# dtype: bool
```

6. `pd.notnull()`

- 用法: `pd.notnull(obj)` - 检测类似数组的对象中的非缺失值。
- 示例 (来自课件 page 17):

```python
s4 = pd.Series({'Number': np.nan, 'Name': 'Zhang San', 'Weight': 178, 'Height': 66})
print(pd.notnull(s4))
# 输出:
# Number    False
# Name       True
# Weight     True
# Height     True
# dtype: bool
```

7. `.isnull()` (Series/DataFrame 方法)

- 用法: `Series.isnull()` 或 `DataFrame.isnull()` - 检测 Series 或 DataFrame 中的缺失值。
- 示例 (来自课件 page 18, Series 方法):

```python
s4 = pd.Series({'Number': np.nan, 'Name': 'Zhang San', 'Weight': 178, 'Height': 66})
print(s4.isnull())
```

8. `.notnull()` (Series/DataFrame 方法)

- 用法: `Series.notnull()` 或 `DataFrame.notnull()` - 检测 Series 或 DataFrame 中的非缺失值。
- 示例 (来自课件 page 18, Series 方法):

```
s4 = pd.Series({'Number': np.nan, 'Name': 'Zhang San', 'Weight': 178, 'Height': 66})
print(s4.notnull())
```

9. `.append()` (Series 方法 - 注意与列表的 append 不同，Series 的 append 返回新对象)
    - 用法: `Series.append(to_append, ignore_index=False, verify_integrity=False)` - 将其他 Series 或类似字典/列表的对象连接到此 Series 的末尾，返回一个新的 Series。
    - 示例 (来自课件 page 20):

```
s4 = pd.Series({'Name': 'Zhang San', 'Weight': 178, 'Height': 66})
s4_appended = s4.append(pd.Series(['Finance', 'Python'], index=['Major', 'Course']))
print(s4_appended)
```

10. `.drop()` (Series/DataFrame 方法)
    - 用法: `Series.drop(labels=None, axis=0, index=None, columns=None, level=None, inplace=False, errors='raise')` - 从 Series/DataFrame 中移除指定的标签。
    - 示例 (来自课件 page 20, Series drop):

```
s4_appended = pd.Series({'Name':'Zhang San', 'Weight':178, 'Height':66, 'Major':'Finance', 'Course':'Python'})
s4_dropped = s4_appended.drop('Number') # 假设 Number 之前存在
# 课件示例是 s4 = s4.drop('Number')
print(s4_dropped)
```

11. `.head()` (Series/DataFrame 方法)
    - 用法: `Series.head(n=5)` 或 `DataFrame.head(n=5)` - 返回前 `n` 行。
    - 示例 (来自课件 page 21, Series head):

```
data = pd.Series(['python', 'java', 'c++', 'c', 'go', 'html'])
print(data.head())    # 默认前5行
print(data.head(3))   # 指定前3行
```

12. `.tail()` (Series/DataFrame 方法)

- 用法: `Series.tail(n=5)` 或 `DataFrame.tail(n=5)` - 返回后 `n` 行。
- 示例 (来自课件 page 21, Series tail):

```
data = pd.Series(['python', 'java', 'c++', 'c',
'go', 'html'])
print(data.tail(3))
```

13. `pd.DataFrame()`

- 用法: `pd.DataFrame(data=None, index=None, columns=None, dtype=None, copy=None)` - 创建一个二维、大小可变、具有标记轴（行和列）的表格数据结构。
- 示例 (来自课件 page 24, 创建空 DataFrame):

```
df = pd.DataFrame()
print(df)
```

- 示例 (来自课件 page 24, 从字典创建 DataFrame):

```
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'],
        'Age': [28, 34, 29, 42],
        'BMI': [17.5, 25.0, 21.0, 22.3]}
df = pd.DataFrame(data)
print(df)
```

14. `pd.read_csv()`

- 用法: `pd.read_csv(filepath_or_buffer, sep=',', delimiter=None, header='infer', names=None, index_col=None, ...)` - 将 CSV（逗号分隔）文件读入 DataFrame。
- 示例 (来自课件 page 26):

```
df_csv = pd.read_csv('../data/my_csv.csv')
print(df_csv.head())
```

- 示例 (来自课件 page 26, 指定 header 和 index_col):

```
# pd.read_csv('../data/my_table.txt', header=None,
index_col=['col1']) # 假设用法
```

15. `pd.read_table()`

- 用法: `pd.read_table(filepath_or_buffer, sep='\t', delimiter=None, header='infer', ...)` - 将通用分隔文件读入 DataFrame。默认分隔符是制表符 `\t`。
- 示例 (来自课件 page 26):

```
df_txt = pd.read_table('../data/my_table.txt')
print(df_txt.head())
```

- 示例 (来自课件 page 27, 使用自定义分隔符):

```
df_special_sep = pd.read_table('../data/my_table_special_sep.txt', sep='\|\|\|\|', engine='python')
print(df_special_sep)
```

16. `pd.read_excel()`

- 用法: `pd.read_excel(io, sheet_name=0, header=0, names=None, index_col=None, ...)` - 将 Excel 文件读入 pandas DataFrame。
- 示例 (来自课件 page 26):

```
df_excel = pd.read_excel('../data/my_excel.xlsx')
print(df_excel.head())
```

17. `.to_csv()` (DataFrame/Series 方法)

- 用法: `DataFrame.to_csv(path_or_buf=None, sep=',', na_rep='', float_format=None, columns=None, header=True, index=True, ...)` - 将对象写入逗号分隔值 (csv) 文件。
- 示例 (来自课件 page 28):

```
df_csv.to_csv('../data/my_csv_saved.csv', index=False)
df_txt.to_csv('../data/my_txt_saved.txt', sep='\t', index=False)
```

18. `.to_excel()` (DataFrame 方法)

- 用法: `DataFrame.to_excel(excel_writer, sheet_name='Sheet1', na_rep='', float_format=None, columns=None, header=True, index=True, ...)` - 将对象写入 Excel 工作表。
- 示例 (来自课件 page 28):

```
df_excel.to_excel('../data/my_excel_saved.xlsx', index=False)
```

19. `.T` (DataFrame 属性)
    - 用法: `DataFrame.T` - 转置 DataFrame 的索引和列。
    - 示例 (来自课件 page 29):

```
data = {'Name': ['Tom', 'Jack', 'Steve', 'Ricky'], 'Age': [28, 34, 29, 42]}
df = pd.DataFrame(data)
print(df.T)
```

20. `.shape` (DataFrame 属性)
    - 用法: `DataFrame.shape` - 返回一个表示 DataFrame 维度的元组 (行数, 列数)。
    - 示例 (来自课件 page 29):

```
df = pd.DataFrame({'A':[1,2,3],'B':[4,5,6]})
print(df.shape) # 输出: (3, 2)
print(df.shape[0]) # 行数, 输出: 3
print(df.shape[1]) # 列数, 输出: 2
```

21. `.size` (DataFrame 属性)
    - 用法: `DataFrame.size` - 返回 DataFrame 中元素的总数 (行数 * 列数)。
    - 示例 (来自课件 page 29):

```
df = pd.DataFrame({'A':[1,2,3],'B':[4,5,6]})
print(df.size) # 输出: 6
```

22. `.columns` (DataFrame 属性)
    - 用法: `DataFrame.columns` - 返回 DataFrame 的列标签。
    - 示例 (来自课件 page 30):

```
df = pd.DataFrame({'Name': ['Tom'], 'Age': [28], 'BMI': [17.5]})
print(df.columns)
# 输出: Index(['Name', 'Age', 'BMI'], dtype='object')
```

23. `.unique()` (Series 方法)

- 用法: `Series.unique()` - 返回 Series 中的唯一值。
- 示例 (来自课件 page 31):

```
df = pd.DataFrame({'School': ['Tsinghua University', 'Peking University', 'Tsinghua University']})
print(df['School'].unique())
# 输出: array(['Tsinghua University', 'Peking University'], dtype=object)
```

24. `.nunique()` (Series 方法)

- 用法: `Series.nunique(dropna=True)` - 返回 Series 中唯一值的数量。
- 示例 (来自课件 page 31):

```
df = pd.DataFrame({'School': ['Tsinghua University', 'Peking University', 'Tsinghua University']})
print(df['School'].nunique())
# 输出: 2
```

25. `.value_counts()` (Series 方法)

- 用法: `Series.value_counts(normalize=False, sort=True, ascending=False, bins=None, dropna=True)` - 返回一个包含唯一值计数的 Series。
- 示例 (来自课件 page 31):

```python
df = pd.DataFrame({'School': ['Tsinghua University', 'Peking University', 'Tsinghua University', 'Fudan University', 'Peking University', 'Tsinghua University']})
print(df['School'].value_counts())
# 输出:
# Tsinghua University    3
# Peking University      2
# Fudan University       1
# Name: School, dtype: int64
```

26. `.drop_duplicates()` (DataFrame/Series 方法)

- 用法: `DataFrame.drop_duplicates(subset=None, keep='first', inplace=False, ignore_index=False)` - 返回删除了重复行的 DataFrame/Series。
- 示例 (来自课件 page 32):

```python
df = pd.DataFrame({'Gender': ['Female', 'Male', 'Female', 'Male'], 'Transfer': ['N', 'N', 'Y', 'N']})
df_demo = df.drop_duplicates(['Gender', 'Transfer'])
print(df_demo)
# 假设 keep='first' (默认)
# 输出 (只保留第一次出现的组合):
#    Gender Transfer
# 0  Female        N
# 1    Male        N
# 2  Female        Y
```

27. `.replace()` (Series/DataFrame 方法)

- 用法: `Series.replace(to_replace=None, value=None, inplace=False, limit=None, regex=False, method='pad')` - 替换 `to_replace` 中的值。
- 示例 (来自课件 page 33, 字典映射替换):

```python
df = pd.DataFrame({'Gender': ['Female', 'Male', 'Female', 'Female', 'Male']})
print(df['Gender'].replace({'Female':0, 'Male':1}).head())
# 输出:
# 0    0
# 1    1
# 2    0
# 3    0
# 4    1
# Name: Gender, dtype: int64
```

- 示例 (来自课件 page 33, 列表替换):

```python
print(df['Gender'].replace(['Female', 'Male'], [0, 1]).head()) # 效果同上
```

28. `.where()` (Series/DataFrame 方法)

- 用法: `Series.where(cond, other=nan, inplace=False, axis=None, level=None, errors='raise', try_cast=False)` - 根据条件替换值。如果条件为 True，则保留原始值；如果为 False，则替换为 `other`。
- 示例 (来自课件 page 34):

```python
s = pd.Series([-1, 1.2345, 100, -50])
print(s.where(s > 0)) # 条件为 True 的保留，False 的变为 NaN
# 输出:
# 0       NaN
# 1    1.2345
# 2   100.0000
# 3       NaN
# dtype: float64
```

29. `.mask()` (Series/DataFrame 方法)

- 用法: `Series.mask(cond, other=nan, inplace=False, axis=None, level=None, errors='raise', try_cast=False)` - 根据条件替换值。如果条件为 True，则替换为 `other`；如果为 False，则保留原始值。
- 示例 (来自课件 page 34):

```python
s = pd.Series([-1, 1.2345, 100, -50])
print(s.mask(s > 0)) # 条件为 True 的变为 NaN，False 的保留
# 输出:
# 0    -1.0
# 1     NaN
# 2     NaN
# 3   -50.0
# dtype: float64
```

30. `.round()` (Series/DataFrame 方法)

- 用法: `Series.round(decimals=0, *args, **kwargs)` - 将 Series 中的每个值四舍五入到给定的小数位数。
- 示例 (来自课件 page 35):

```python
s = pd.Series([-1, 1.2345, 100, -50])
print(s.round(2))
# 输出:
# 0     -1.00
# 1      1.23
# 2    100.00
# 3    -50.00
# dtype: float64
```

31. `.abs()` (Series/DataFrame 方法)

- 用法: `Series.abs()` - 返回包含每个元素的绝对数值的新 Series/DataFrame。
- 示例 (来自课件 page 35):

```python
s = pd.Series([-1, 1.2345, 100, -50])
print(s.abs())
# 输出:
# 0       1.0000
# 1       1.2345
# 2     100.0000
# 3      50.0000
# dtype: float64
```

32. `.clip()` (Series/DataFrame 方法)

- 用法: `Series.clip(lower=None, upper=None, axis=None, inplace=False, *args, **kwargs)` - 将值修剪到输入的阈值。
- 示例 (来自课件 page 35):

```python
s = pd.Series([-1, 1.2345, 100, -50])
print(s.clip(0, 2)) # 将小于0的值设为0，大于2的值设为2
# 输出:
# 0    0.0000
# 1    1.2345
# 2    2.0000
# 3    0.0000
# dtype: float64
```

33. `.info()` (DataFrame 方法)

- 用法: `DataFrame.info(verbose=None, buf=None, max_cols=None, memory_usage=None, show_counts=None, null_counts=None)` - 打印 DataFrame 的简明摘要。
- 示例 (来自课件 page 36):

```python
# 假设 df 是一个已加载的 DataFrame
# df.info()
# 输出类似:
# <class 'pandas.core.frame.DataFrame'>
# RangeIndex: 200 entries, 0 to 199
# Data columns (total 7 columns):
#  #   Column    Non-Null Count  Dtype
# ---  ------    --------------  -----
#  0   School    200 non-null    object
# ...
# memory usage: 11.1+ KB
```

34. `.describe()` (DataFrame/Series 方法)

- 用法: `DataFrame.describe(percentiles=None, include=None, exclude=None, datetime_is_numeric=False)` - 生成描述性统计数据。
- 示例 (来自课件 page 36):

```
# 假设 df 是一个包含数值列的 DataFrame
# print(df.describe())
# 输出类似:
#              Height       Weight
# count   200.000000   200.000000
# mean    163.218033    55.015873
# std       8.608879    12.824294
# min     145.400000    34.000000
# 25%     157.150000    46.000000
# 50%     161.900000    51.000000
# 75%     167.500000    65.000000
# max     193.900000    89.000000
```

35. `.sort_values()` (DataFrame/Series 方法)

- 用法: `DataFrame.sort_values(by, axis=0, ascending=True, inplace=False, kind='quicksort', na_position='last', ignore_index=False, key=None)` - 按一个或多个列/行中的值排序。
- 示例 (来自课件 page 38):

```
# 假设 df_demo 是一个 DataFrame
# print(df_demo.sort_values(['Weight', 'Height'], ascending=[True, False]).head())
# 按 Weight 升序，然后按 Height 降序
```

36. `.sort_index()` (DataFrame/Series 方法)

- 用法: `DataFrame.sort_index(axis=0, level=None, ascending=True, inplace=False, kind='quicksort', na_position='last', sort_remaining=True, ignore_index=False, key=None)` - 按索引标签排序。
- 示例 (来自课件 page 38, 旁边提到了但未直接展示 .sort_index() 的输出，通常用于多级索引排序或确保索引有序):

```
# df_multi.sort_index(inplace=True) # 对多级索引排序
```

37. `.apply()` (DataFrame/Series 方法)

- 用法: `DataFrame.apply(func, axis=0, raw=False, result_type=None, args=(), **kwargs)` - 将函数应用于轴上的一个或多个元素。
- 示例 (来自课件 page 39):

```python
df_demo = pd.DataFrame({'Height': [163.2, 189.0], 'Weight': [55.0, 89.0]})
def my_mean(x): # x 在这里是一个 Series (一列)
    res = x.mean()
    return res
print(df_demo.apply(my_mean))
# 输出:
# Height    176.10
# Weight     72.00
# dtype: float64
```

38. `.rolling()` (DataFrame/Series 方法)

- 用法: `DataFrame.rolling(window, min_periods=None, center=False, win_type=None, on=None, axis=0, closed=None, step=None, method='single')` - 提供滚动窗口计算。
- 示例 (来自课件 page 40):

```python
s = pd.Series([1, 2, 3, 4, 5])
roller = s.rolling(window = 3)
print(roller) # 输出一个 Rolling 对象
print(roller.mean())
# 输出:
# 0    NaN
# 1    NaN
# 2    2.0  (1+2+3)/3
# 3    3.0  (2+3+4)/3
# 4    4.0  (3+4+5)/3
# dtype: float64
```

39. `.shift()` (DataFrame/Series 方法)

- 用法: `DataFrame.shift(periods=1, freq=None, axis=0, fill_value=None)` - 将索引移动指定的期数。
- 示例 (来自课件 page 41):

```
s = pd.Series([1,3,6,10,15])
print(s.shift(2)) # 向下移动2位，前面用NaN填充
# 输出:
# 0    NaN
# 1    NaN
# 2    1.0
# 3    3.0
# 4    6.0
# dtype: float64
```

40. `.diff()` (DataFrame/Series 方法)
    - 用法: `DataFrame.diff(periods=1, axis=0)` - 元素的一阶离散差分。
    - 示例 (来自课件 page 41):

```
s = pd.Series([1,3,6,10,15])
print(s.diff(1)) # 与前一个元素做差
# 输出:
# 0    NaN
# 1    2.0  (3-1)
# 2    3.0  (6-3)
# 3    4.0  (10-6)
# 4    5.0  (15-10)
# dtype: float64
```

41. `.pct_change()` (DataFrame/Series 方法)
    - 用法: `DataFrame.pct_change(periods=1, fill_method='pad', limit=None, freq=None, **kwargs)` - 元素之间的百分比变化。
    - 示例 (来自课件 page 41):

```
s = pd.Series([1,3,6,10,15])
print(s.pct_change()) # (当前-前一个)/前一个
# 输出:
# 0         NaN
# 1    2.000000  ( (3-1)/1 )
# 2    1.000000  ( (6-3)/3 )
# 3    0.666667  ( (10-6)/6 )
# 4    0.500000  ( (15-10)/10 )
# dtype: float64
```

42. `.expanding()` (DataFrame/Series 方法)

- 用法: `DataFrame.expanding(min_periods=1, center=None, axis=0, method='single')` - 提供扩展窗口计算。
- 示例 (来自课件 page 42):

```python
s = pd.Series([1, 3, 6, 10])
print(s.expanding().mean())
# 输出:
# 0     1.000000  (1)/1
# 1     2.000000  (1+3)/2
# 2     3.333333  (1+3+6)/3
# 3     5.000000  (1+3+6+10)/4
# dtype: float64
```

43. `.loc[]` (DataFrame/Series 索引器)

- 用法: `DataFrame.loc[row_indexer, column_indexer]` - 通过标签或布尔数组访问一组行和列。
- 示例 (来自课件 page 45, 单个元素行索引):

```python
# 假设 df_demo 是一个 DataFrame 且 Name 列已设为索引
# print(df_demo.loc['Qiang Sun']) # 选取 Name 为 'Qiang Sun' 的行
```

- 示例 (来自课件 page 46, 元素列表行索引和列索引):

```python
# print(df_demo.loc[['Qiang Sun', 'Quan Zhao'], ['School', 'Gender']])
```

44. `.iloc[]` (DataFrame/Series 索引器)

- 用法: `DataFrame.iloc[row_indexer, column_indexer]` - 主要基于整数位置进行索引（从 0 到 length-1 的轴）。
- 示例 (来自课件 page 50):

```python
# 假设 df_demo 是一个 DataFrame
# print(df_demo.iloc[1:4, 2:4]) # 选取行1到3，列2到3 (切片不包含结束点)
# 输出:
#     Gender Weight
# 1     Male   89.0
# 2   Female   41.0
# 3     Male   73.0
```

45. `.query()` (DataFrame 方法)

- 用法: `DataFrame.query(expr, inplace=False, **kwargs)` - 使用布尔表达式查询 DataFrame 的列。
- 示例 (来自课件 page 51):

```
# 假设 df_query 是一个 DataFrame
# print(df_query.query("(School == 'Fudan University' & Grade == 'Senior' & Weight > 70) | (School == 'Peking University' & Grade == 'Senior' & Weight > 80)"))
# 也可写为:
# print(df_query.query("Age > 17 & Age < 22"))
```

46. `.set_index()` (DataFrame 方法)

- 用法: `DataFrame.set_index(keys, drop=True, append=False, inplace=False, verify_integrity=False)` - 使用现有列设置 DataFrame 索引。
- 示例 (来自课件 page 44, 为 loc 示例准备):

```
# df_demo = df.set_index('Name')
```

- 示例 (来自课件 page 52, 设置多级索引):

```
# df_new.set_index('A', append=True) # 将 A 列追加到现有索引
```

47. `.reindex()` (DataFrame/Series 方法)

- 用法: `DataFrame.reindex(labels=None, index=None, columns=None, axis=None, method=None, copy=True, level=None, fill_value=nan, limit=None, tolerance=None)` - 使 DataFrame/Series 符合新的索引，并可选择填充逻辑。
- 示例 (来自课件 page 53):

```
df_reindex = pd.DataFrame({'Weight':
[60,70,80],'Height':[176,180,179]},index=
['1001','1003','1002'])
print(df_reindex.reindex(index=
['1001','1002','1003','1004'], columns=
['Weight','Gender']))
# 输出:
#       Weight  Gender
# 1001    60.0     NaN
# 1002    80.0     NaN
# 1003    70.0     NaN
# 1004     NaN     NaN
```

48. `.sample()` (DataFrame/Series 方法)

- 用法: `DataFrame.sample(n=None, frac=None, replace=False, weights=None, random_state=None, axis=None, ignore_index=False)` - 从对象轴返回项目的随机样本。
- 示例 (来自课件 page 55, 随机抽取3个样本，有放回，按 value 列的权重):

```
df_sample = pd.DataFrame({'id': list('abcde'),
'value': [1, 2, 3, 4, 90]})
print(df_sample.sample(3, replace=True,
weights=df_sample.value))
```

49. `pd.MultiIndex.from_product()`

- 用法: `pd.MultiIndex.from_product(iterables, sortorder=None, names=None)` - 从可迭代对象的笛卡尔积创建一个 MultiIndex。
- 示例 (来自课件 page 56):

```
multi_index =
pd.MultiIndex.from_product([list('ABCD'), ['Male',
'Female']], names=('School', 'Gender'))
print(multi_index)
```

50. `.sort_index()` (用于多级索引, DataFrame/Series 方法)

- 用法: `DataFrame.sort_index(axis=0, level=None, ascending=True, inplace=False, kind='quicksort', na_position='last', sort_remaining=True, ignore_index=False, key=None)` - 按索引标签排序。对于多级索引，可以指定 `level`。

- 示例 (来自课件 page 60, 为了使用切片对多级索引排序):

```
# df_multi 是一个具有多级索引的 DataFrame
# df_sorted = df_multi.sort_index()
# print(df_sorted.loc[('Fudan University', 'Senior'):].head()) # 排序后才能用切片
```

pandas高级操作-数据处理 (File 10)

(这个文件中的函数在上面的 `pandas的使用` 文件中大部分已经覆盖，这里列出一些强调的用法或新的组合)

1. `.groupby()` (DataFrame/Series 方法)
    - 用法: `DataFrame.groupby(by=None, axis=0, level=None, as_index=True, sort=True, group_keys=True, squeeze=<no_default>, observed=False, dropna=True)` - 使用映射器或按一系列列对 DataFrame 进行分组。
    - 示例 (来自课件 page 2, 单列分组求均值):

```
# 假设 df 是一个 DataFrame
# print(df.groupby('Gender')['Longevity'].mean())
```

    - 示例 (来自课件 page 2, 按布尔条件分组):

```
# df = pd.read_csv('../data/learn_pandas.csv')
# condition = df.Gender == 'Male' # 假设 Gender 列存在
# print(df.groupby(condition)['Height'].median())
```

    - 示例 (来自课件 page 3, 多列分组):

```
# print(df.groupby(['School', 'Gender'])['Height'].mean())
```

2. `.ngroups` (Groupby 对象属性)
    - 用法: `groupby_object.ngroups` - 返回分组的数量。
    - 示例 (来自课件 page 5):

```
# gb = df.groupby(['School', 'Grade'])
# print(gb.ngroups) # 输出分组数量
```

3. `.groups` (Groupby 对象属性)

- 用法: `groupby_object.groups` - 返回一个字典，其键是计算出的唯一组，对应的值是属于每个组的轴标签。
- 示例 (来自课件 page 6):

```
# res = gb.groups
# print(res.keys()) # 打印所有分组的键 (元组形式)
```

4. `.agg()` (Groupby/DataFrame/Series 方法)

- 用法: `groupby_object.agg(func=None, *args, **kwargs)` 或 `DataFrame.agg(func=None, axis=0, *args, **kwargs)` - 使用指定轴上的一个或多个操作进行聚合。
- 示例 (来自课件 page 10, 对分组应用多个聚合函数):

```
# gb = df.groupby('Gender')
# print(gb.agg(['sum', 'idxmax', 'skew']))
```

- 示例 (来自课件 page 11, 对不同列应用不同聚合函数):

```
# print(gb.agg({'Height': ['mean', 'max'],
'Weight': 'count'}))
```

- 示例 (来自课件 page 12, 使用自定义 lambda 函数):

```
# print(gb.agg(lambda x: x.mean() - x.min()))
```

- 示例 (来自课件 page 13, 聚合结果重命名):

```
# print(gb.agg([('range', lambda x: x.max() -
x.min()), ('my_sum', 'sum')]))
# 或对特定列重命名
# print(gb.agg({'Height': [('my_func', 'my_sum'),
'sum'], 'Weight': lambda x:x.max()}))
```

5. `.merge()` (DataFrame 方法 / pandas顶级函数)

- 用法: `pd.merge(left, right, how='inner', on=None, left_on=None, right_on=None, left_index=False, right_index=False, sort=False, suffixes=('_x', '_y'), copy=True, indicator=False, validate=None)` - 通过数据库风格的连接合并 DataFrame 或命名 Series 对象。
- 示例 (来自课件 page 15, 左连接):

```
# df1 = pd.DataFrame({'Name':['San Zhang', 'Si Li'], 'Age':[20,30]})
# df2 = pd.DataFrame({'Name':['Si Li', 'Wu Wang'], 'Gender':['F', 'M']})
# print(df1.merge(df2, on='Name', how='left'))
```

- 示例 (来自课件 page 16, 使用 `left_on` 和 `right_on`):

```
# df1 = pd.DataFrame({'df1_name':['San Zhang', 'Si Li'], 'Age':[20,30]})
# df2 = pd.DataFrame({'df2_name':['Si Li', 'Wu Wang'], 'Gender':['F', 'M']})
# print(df1.merge(df2, left_on='df1_name', right_on='df2_name', how='left'))
```

6. `.join()` (DataFrame 方法)
   - 用法: `DataFrame.join(other, on=None, how='left', lsuffix='', rsuffix='', sort=False, validate=None)` - 使用其索引或基于键列连接另一个 DataFrame 的列。
   - 示例 (来自课件 page 17, 索引连接):

```
# df1 = pd.DataFrame({'Age':[20,30]}, index=pd.Series(['San Zhang', 'Si Li'], name='Name'))
# df2 = pd.DataFrame({'Gender':['F', 'M']}, index=pd.Series(['Si Li', 'Wu Wang'], name='Name'))
# print(df1.join(df2, how='left'))
```

7. `pd.concat()` (pandas顶级函数)
   - 用法: `pd.concat(objs, axis=0, join='outer', ignore_index=False, keys=None, levels=None, names=None, verify_integrity=False, sort=False, copy=True)` - 沿特定轴连接 pandas 对象。

- 示例 (来自课件 page 18, 纵向连接):

```
# df1 = pd.DataFrame({'Name':['San Zhang', 'Si Li'], 'Age':[20,30]})
# df2 = pd.DataFrame({'Name':['Wu Wang'], 'Age':[40]})
# print(pd.concat([df1, df2])) # 默认 axis=0
```

- 示例 (来自课件 page 19, 横向连接):

```
# df2 = pd.DataFrame({'Grade':[80,90]}) # 假设与 df1 行数匹配
# df3 = pd.DataFrame({'Gender':['M','F']})
# print(pd.concat([df1, df2, df3], axis=1))
```

8. `.pivot()` (DataFrame 方法)

- 用法: `DataFrame.pivot(index=None, columns=None, values=None)` - 返回根据给定索引/列值重塑的 DataFrame。要求索引和列的组合是唯一的。
- 示例 (来自课件 page 22):

```
# df = pd.DataFrame({'Class':[1,1,2,2], 'Name':['San Zhang','San Zhang','Si Li','Si Li'], 'Subject':['Chinese','Math','Chinese','Math'], 'Grade':[80,75,90,85]})
# print(df.pivot(index='Name', columns='Subject', values='Grade'))
# 输出:
# Subject    Chinese  Math
# Name
# San Zhang       80    75
# Si Li           90    85
```

9. `.pivot_table()` (DataFrame 方法 / pandas顶级函数)

- 用法: `pd.pivot_table(data, values=None, index=None, columns=None, aggfunc='mean', fill_value=None, margins=False, dropna=True, margins_name='All', observed=False, sort=True)` - 创建一个类似电子表格数据透视表作为 DataFrame。
- 示例 (来自课件 page 25, 处理重复值):

```
# df = pd.DataFrame({'Name':['San Zhang','San Zhang','Si Li','Si Li'], 'Subject':['Chinese','Chinese','Math','Math'], 'Grade':[80,90,70,80]})
# print(pd.pivot_table(df, index='Name', columns='Subject', values='Grade', aggfunc='mean'))
# 输出:
# Subject    Chinese  Math
# Name
# San Zhang     85.0   NaN
# Si Li          NaN  75.0
```

- 示例 (来自课件 page 26, 带边际汇总):

```
# print(pd.pivot_table(df, index='Name', columns='Subject', values='Grade', aggfunc='mean', margins=True))
```

10. `.melt()` (DataFrame 方法 / pandas顶级函数)

- 用法: `pd.melt(frame, id_vars=None, value_vars=None, var_name=None, value_name='value', col_level=None, ignore_index=True)` - "Unpivot" a DataFrame from wide to long format.
- 示例 (来自课件 page 27):

```
# df_wide = pd.DataFrame({'Class':[0,1], 'Name':['San Zhang','Si Li'], 'Chinese':[80,90], 'Math':[80,75]})
# df_melted = pd.melt(df_wide, id_vars=['Class', 'Name'], value_vars=['Chinese', 'Math'], var_name='Subject', value_name='Grade')
# print(df_melted)
```

11. `.unstack()` (Series/DataFrame 方法)

- 用法: `Series.unstack(level=-1, fill_value=None)` - 将具有多级索引（MultiIndex）的 Series 转换为 DataFrame，或将 DataFrame 的行索引级别提升为列轴。
- 示例 (来自课件 page 28):

```
# 假设 df 是一个具有多级行索引的 DataFrame
# print(df.unstack()) # 默认 unstack 最内层行索引
# print(df.unstack(level=0)) # unstack 指定层级的行索引
# print(df.unstack([0,2])) # unstack 多个层级
```

12. `.stack()` (DataFrame 方法)
    - 用法: `DataFrame.stack(level=-1, dropna=True)` - 将 DataFrame 的列级别堆叠到索引中，返回一个具有新内层行索引的 Series 或 DataFrame。
    - 示例 (来自课件 page 30):

```
# 假设 df_unstacked 是一个具有列索引的 DataFrame
# print(df_unstacked.stack()) # 默认 stack 最内层列索引
# print(df_unstacked.stack(level=[1,2])) # stack 指定层级的列索引
```

13. `.str` (Series 访问器)
    - 用法: `Series.str` - 访问 Series 中每个元素的字符串方法。
    - 示例 (来自课件 page 31):

```
# s = pd.Series(['abcd', 'efg', 'hi'])
# print(s.str.upper())
# 输出:
# 0    ABCD
# 1     EFG
# 2      HI
# dtype: object
```

14. `.str.split()` (Series 字符串方法)
    - 用法: `Series.str.split(pat=None, n=-1, expand=False, regex=None)` - 在给定分隔符/定界符周围拆分字符串。
    - 示例 (来自课件 page 33):

```
# s = pd.Series(['上海市 黄浦区 方浜中路249号', '上海市 宝山区 密山路5号'])
# print(s.str.split('[市区路]', n=2, expand=True))
# 输出 DataFrame:
#       0     1          2
# 0   上海   黄浦   方浜中路249号
# 1   上海   宝山     密山路5号
```

15. `.str.join()` (Series 字符串方法)
    - 用法: `Series.str.join(sep)` - 使用传递的定界符连接 Series 中每个元素的列表。
    - 示例 (来自课件 page 34):

```
# s = pd.Series([['a','b'], ['c','d'], np.nan, [['e','f'],['g','h']]]) # 假设数据
# print(s.str.join('-'))
# 输出:
# 0     a-b
# 1     c-d
# 2     NaN
# 3     NaN # 因为内部是列表的列表，join期望元素是字符串
#
# 正确的例子:
s = pd.Series([['a','b'], ['c','d']])
print(s.str.join('-'))
# 输出:
# 0    a-b
# 1    c-d
# dtype: object
```

16. `.str.cat()` (Series 字符串方法)
    - 用法: `Series.str.cat(others=None, sep=None, na_rep=None, join='left')` - 将 Series/Index 中的字符串与另一个 Series/Index/list-like 连接起来。
    - 示例 (来自课件 page 35):

```
# s1 = pd.Series(['a','b'])
# s2 = pd.Series(['cat','dog'])
# print(s1.str.cat(s2, sep='-'))
# 输出:
# 0    a-cat
# 1    b-dog
# dtype: object
```

17. `.str.contains()` (Series 字符串方法)
    - 用法: `Series.str.contains(pat, case=True, flags=0, na=nan, regex=True)` - 测试模式或正则表达式是否包含在 Series 或 Index 的字符串中。
    - 示例 (来自课件 page 36):

```
# s = pd.Series(['my cat', 'he is fat', 'railway station'])
# print(s.str.contains('\s(wat)')) # 查找包含 " wat" (空格后跟wat)
# 输出:
# 0    False
# 1    False
# 2     True
# dtype: bool
```

18. `.str.startswith()` (Series 字符串方法)
    - 用法: `Series.str.startswith(pat, na=nan)` - 测试 Series 或 Index 中每个字符串的开头是否与模式匹配。
    - 示例 (来自课件 page 36):

```
# s = pd.Series(['my cat', 'he is fat', 'railway station'])
# print(s.str.startswith('my'))
# 输出:
# 0     True
# 1    False
# 2    False
# dtype: bool
```

19. `.str.endswith()` (Series 字符串方法)

- 用法: `Series.str.endswith(pat, na=nan)` - 测试 Series 或 Index 中每个字符串的结尾是否与模式匹配。
- 示例 (来自课件 page 36):

```
# s = pd.Series(['my cat', 'he is fat', 'railway station'])
# print(s.str.endswith('t'))
# 输出:
# 0     True
# 1     True
# 2    False
# dtype: bool
```

20. `.str.match()` (Series 字符串方法)

- 用法: `Series.str.match(pat, case=True, flags=0, na=nan)` - 确定 Series/Index 中的每个字符串是否与正则表达式匹配。
- 示例 (来自课件 page 37):

```
# s = pd.Series(['m cat', 'h is fat', 'railway station'])
# print(s.str.match('(m|h)')) # 匹配以 m 或 h 开头的字符串
# 输出:
# 0     True
# 1     True
# 2    False
# dtype: bool
```

21. `.str.find()` (Series 字符串方法)

- 用法: `Series.str.find(sub, start=0, end=None)` - 返回 Series/Index 中每个字符串中子字符串第一次出现的最低索引。
- 示例 (来自课件 page 38):

```
# s = pd.Series(['This is an apple. That is not an apple.'])
# print(s.str.find('apple'))
# 输出:
# 0    11
# dtype: int64
```

22. `.str.rfind()` (Series 字符串方法)

- 用法: `Series.str.rfind(sub, start=0, end=None)` - 返回 Series/Index 中每个字符串中子字符串最后一次出现的最高索引。
- 示例 (来自课件 page 38):

```
# s = pd.Series(['This is an apple. That is not an apple.'])
# print(s.str.rfind('apple'))
# 输出:
# 0    33
# dtype: int64
```

23. `.str.replace()` (Series 字符串方法)

- 用法: `Series.str.replace(pat, repl, n=-1, case=None, flags=0, regex=None)` - 替换 Series/Index 中每次出现的模式/正则表达式。
- 示例 (来自课件 page 39):

```
# s = pd.Series(['a_1_b', 'c_?'])
# print(s.str.replace('(_\d|\?)', 'new', regex=True))
# 输出:
# 0    anew_b  (替换 _1)
# 1    cnew    (替换 _?)
# dtype: object
```

24. `.str.extract()` (Series 字符串方法)

- 用法: `Series.str.extract(pat, flags=0, expand=True)` - 从 Series/Index 中的每个字符串中提取捕获组作为 DataFrame 中的列。
- 示例 (来自课件 page 40):

```
# pat = '(\w+市)(\w+区)(\w+路)(\d+号)' # 市、区、路、号
# s = pd.Series(['上海市黄浦区方浜中路249号', '北京市昌平区北农路2号'])
# print(s.str.extract(pat))
# 输出 DataFrame:
#       0    1       2     3
# 0  上海市  黄浦区  方浜中路  249号
# 1  北京市  昌平区   北农路    2号
```

25. `.str.extractall()` (Series 字符串方法)

- 用法: `Series.str.extractall(pat, flags=0)` - 从 Series/Index 中的每个字符串中提取所有出现的捕获组，返回一个 MultiIndex DataFrame。
- 示例 (来自课件 page 42):

```
# s = pd.Series(["A13ST15, A26S", "B674S2, B25T0"], index=["my_A", "my_B"])
# pat = r"([A|B])(\d+)([S|T])(\d+)"
# print(s.str.extractall(pat))
# 输出 MultiIndex DataFrame
```

26. `.boxplot()` (DataFrame/Series 方法)
    - 用法: `DataFrame.boxplot(column=None, by=None, ax=None, fontsize=None, rot=0, grid=True, figsize=None, layout=None, return_type=None, **kwargs)` - 从 DataFrame 列制作箱形图。
    - 示例 (来自课件 page 46, Kaggle 项目):

```
# data.boxplot(column='Attack', by='Legendary')
# plt.show() # 需要配合 matplotlib.pyplot
```

27. `.corr()` (DataFrame/Series 方法)
    - 用法: `DataFrame.corr(method='pearson', min_periods=1, numeric_only=False)` - 计算列的成对相关性，不包括 NA/null 值。
    - 示例 (来自课件 page 44, Kaggle 项目):

```
# print(data.corr())
```

Python 与大数据分析 --数据可视化基础 (File 12)

(这部分主要是 Matplotlib 和 Seaborn 的绘图函数，很多在上面已经出现过，这里主要关注其绘图相关的特定用法。)

1. `import matplotlib.pyplot as plt`
    - 用法: 导入 Matplotlib 的 pyplot 模块，通常别名为 `plt`。
    - 示例 (来自课件 page 12):

```
import matplotlib.pyplot as plt
import numpy as np
```

2. `plt.figure()`

- 用法: `plt.figure(num=None, figsize=None, dpi=None, facecolor=None, edgecolor=None, frameon=True, FigureClass=<class 'matplotlib.figure.Figure'>, clear=False, **kwargs)` - 创建一个新的图形，或激活一个现有的图形。
- 示例 (来自课件 page 11):

```
plt.figure(figsize=(8, 4)) # 创建一个8x4英寸的图形
```

3. `plt.plot()`

- 用法: `plt.plot(*args, scalex=True, scaley=True, data=None, **kwargs)` - 绘制 y 对 x 的线和/或标记。
- 示例 (来自课件 page 12, 绘制正弦曲线):

```
x = np.arange(0, 2 * np.pi, 0.1)
y = np.sin(x)
plt.plot(x, y)
plt.show()
```

- 示例 (来自课件 page 14, 带标签、颜色、线宽):

```
plt.plot(x, y, label="$sin(x)$", color="red", linewidth=2)
```

4. `plt.show()`

- 用法: `plt.show(*args, **kwargs)` - 显示所有打开的图形。
- 示例 (来自课件 page 12):

```
plt.show()
```

5. `plt.legend()`

- 用法: `plt.legend(*args, **kwargs)` - 在轴上放置图例。
- 示例 (来自课件 page 14):

```
plt.legend()
```

- 示例 (来自课件 page 26, 指定位置和边框):

```
plt.legend(loc='lower right', frameon=False, bbox_to_anchor=(0.5,-0.3))
```

6. `plt.xlabel()`

   - 用法: `plt.xlabel(xlabel, fontdict=None, labelpad=None, **kwargs)` - 设置 x 轴的标签。
   - 示例 (来自课件 page 14):

```
plt.xlabel("x")
```

7. `plt.ylabel()`

   - 用法: `plt.ylabel(ylabel, fontdict=None, labelpad=None, **kwargs)` - 设置 y 轴的标签。
   - 示例 (来自课件 page 14):

```
plt.ylabel("y")
```

8. `plt.title()`

   - 用法: `plt.title(label, fontdict=None, loc=None, pad=None, **kwargs)` - 设置当前轴的标题。
   - 示例 (来自课件 page 17):

```
plt.title('Square Numbers', fontsize=24)
```

9. `plt.savefig()`

   - 用法: `plt.savefig(fname, dpi=None, facecolor='w', edgecolor='w', orientation='portrait', papertype=None, format=None, transparent=False, bbox_inches=None, pad_inches=0.1, frameon=None, metadata=None)` - 将当前图形保存到文件。
   - 示例 (来自课件 page 15):

```
plt.savefig("test.png", dpi=120)
```

10. `plt.scatter()`

    - 用法: `plt.scatter(x, y, s=None, c=None, marker=None, cmap=None, norm=None, vmin=None, vmax=None, alpha=None, linewidths=None, *, edgecolors=None, plotnonfinite=False, data=None, **kwargs)` - 制作 `y` 对 `x` 的散点图。

- 示例 (来自课件 page 16):

```
x=[1,2]
y=[2,4]
plt.scatter(x,y)
```

- 示例 (来自课件 page 18, 指定大小和透明度):

```
plt.scatter(xr, yr, s=area, alpha=0.5)
```

11. `plt.tick_params()`

- 用法: `plt.tick_params(axis='both', **kwargs)` - 更改刻度、刻度标签和网格线的外观。
- 示例 (来自课件 page 17):

```
plt.tick_params(axis='both', which='major',
labelsize=14)
```

12. `plt.rcParams[]` (Matplotlib 全局参数配置)

- 用法: `plt.rcParams['parameter_name'] = value` - 用于设置 Matplotlib 的全局参数，如字体、字号等。
- 示例 (来自课件 page 22, 设置中文字体和负号显示):

```
plt.rcParams['font.sans-serif'] = ['SimHei'] # 用来
正常显示中文标签
plt.rcParams['axes.unicode_minus'] = False # 用来正
常显示负号
```

13. `plt.xticks()`

- 用法: `plt.xticks(ticks=None, labels=None, **kwargs)` - 获取或设置 x 轴的当前刻度位置和标签。
- 示例 (来自课件 page 27):

```
plt.xticks([-np.pi, -np.pi/2, 0, np.pi/2, np.pi])
```

14. `plt.yticks()`

- 用法: `plt.yticks(ticks=None, labels=None, **kwargs)` - 获取或设置 y 轴的当前刻度位置和标签。
- 示例 (来自课件 page 27):

```
plt.yticks([-1, 0, +1])
```

15. `plt.gcf()`

- 用法: `plt.gcf()` - 获取当前图形（Get Current Figure）。
- 示例 (来自课件 page 28):

```
fig = plt.gcf()
```

16. `fig.set_size_inches()` (Figure 对象方法)

- 用法: `figure_object.set_size_inches(w, h, forward=True)` - 以英寸为单位设置图形的宽度和高度。
- 示例 (来自课件 page 28):

```
fig = plt.gcf()
fig.set_size_inches(8.5, 3.5)
```

17. `plt.grid()`

- 用法: `plt.grid(visible=None, which='major', axis='both', **kwargs)` - 配置网格线。
- 示例 (来自课件 page 28):

```
plt.grid() # 显示网格
```

18. `plt.subplot()`

- 用法: `plt.subplot(nrows, ncols, index, **kwargs)` 或 `plt.subplot(pos, **kwargs)` - 在当前图形中添加一个子图。
- 示例 (来自课件 page 29):

```
plt.subplot(221) # 创建 2x2 网格的第一个子图
```

19. `plt.subplots_adjust()`

- 用法: `plt.subplots_adjust(left=None, bottom=None, right=None, top=None, wspace=None, hspace=None)` - 调整子图布局参数。
- 示例 (来自课件 page 30):

```
plt.subplots_adjust(hspace=0.4, wspace=0.3) # 调整子图间的垂直和水平间距
```

20. `plt.bar()`

- 用法: `plt.bar(x, height, width=0.8, bottom=None, *, align='center', data=None, **kwargs)` - 制作条形图。
- 示例 (来自课件 page 35):

```
plt.bar(x=0, height=1)
```

- 示例 (来自课件 page 35, 多个条形):

```
plt.bar(x=(0,1), height=(1,0.5))
```

21. `plt.hist()`

- 用法: `plt.hist(x, bins=None, range=None, density=False, weights=None, cumulative=False, bottom=None, histtype='bar', align='mid', orientation='vertical', rwidth=None, log=False, color=None, label=None, stacked=False, *, data=None, **kwargs)` - 计算并绘制直方图。
- 示例 (来自课件 page 40):

```
# 假设 housing_df 是一个 DataFrame
# plt.hist(housing_df['Avg. Area Number of Rooms'])
```

22. `plt.pie()`

- 用法: `plt.pie(x, explode=None, labels=None, colors=None, autopct=None, pctdistance=0.6, shadow=False, labeldistance=1.1, startangle=0, radius=1, counterclock=True, wedgeprops=None, textprops=None, center=(0, 0), frame=False, rotatelabels=False, *, normalize=True, data=None)` - 绘制饼图。
- 示例 (来自课件 page 43):

```
edu = [0.2515, 0.3724, 0.3336, 0.0368, 0.0057]
labels = ['中专','大专','本科','硕士','其他']
plt.pie(x=edu, labels=labels, autopct='%.1f%%')
```

23. `import seaborn as sns` (导入模块语句)

- 用法: 导入 Seaborn 库，通常别名为 `sns`。
- 示例 (来自课件 page 47):

```
import seaborn as sns
```

24. `sns.set()`
    - 用法: `sns.set(context='notebook', style='darkgrid', palette='deep', font='sans-serif', font_scale=1, color_codes=True, rc=None)` - 设置 Seaborn 绘图的美学参数。
    - 示例 (来自课件 page 47):

```
sns.set() # 应用 Seaborn 默认样式
```

25. `sns.heatmap()`
    - 用法: `sns.heatmap(data, vmin=None, vmax=None, cmap=None, center=None, robust=False, annot=None, fmt='.2g', annot_kws=None, linewidths=0, linecolor='white', cbar=True, cbar_kws=None, cbar_ax=None, square=False, xticklabels='auto', yticklabels='auto', mask=None, ax=None, **kwargs)` - 将矩形数据绘制为颜色编码的矩阵。
    - 示例 (来自课件 page 47):

```
uniform_data = np.random.rand(10, 12)
ax = sns.heatmap(uniform_data)
plt.show()
```

    - 示例 (来自课件 page 48, 指定颜色映射 cmap):

```
ax = sns.heatmap(uniform_data, cmap='YlGnBu')
```

Python 与大数据分析 --机器学习相关函数 (主要来自线性回归、聚类、数据分析文件)

(这部分函数来自 `sklearn` 库)

1. `from sklearn.linear_model import LinearRegression` (导入类)
    - 用法: 导入线性回归模型类。
    - 示例 (来自线性回归课件 page 15):

```
from sklearn.linear_model import LinearRegression
model = LinearRegression()
```

2. `LinearRegression()` (类实例化)

- 用法: 创建 `LinearRegression` 模型对象。
- 示例 (来自线性回归课件 page 15):

```
model = LinearRegression()
```

3. `model.fit()` (模型方法)

- 用法: `model.fit(X_train, y_train)` - 使用训练数据 `X_train` (特征) 和 `y_train` (目标值) 来训练模型。
- 示例 (来自线性回归课件 page 24):

```
# 假设 X_train, y_train 已准备好
# model.fit(X_train, y_train)
```

4. `model.predict()` (模型方法)

- 用法: `model.predict(X_test)` - 使用训练好的模型对新数据 `X_test` 进行预测。
- 示例 (来自线性回归课件 page 25):

```
# y_pred = model.predict(X_test)
# print(y_pred)
```

5. `model.coef_` (模型属性)

- 用法: `model.coef_` - 返回线性回归模型的系数（斜率）。
- 示例 (来自线性回归课件 page 25):

```
# print('回归系数:', model.coef_)
```

6. `model.intercept_` (模型属性)

- 用法: `model.intercept_` - 返回线性回归模型的截距。
- 示例 (来自线性回归课件 page 25):

```
# print('截距:', model.intercept_)
```

7. `from sklearn.model_selection import train_test_split` (导入函数)

- 用法: 导入用于分割数据集的函数。

- 示例 (来自线性回归课件 page 17):

```
from sklearn.model_selection import
train_test_split
```

8. `train_test_split()`
    - 用法: `train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)` - 将数组或矩阵分割成随机的训练和测试子集。
    - 示例 (来自线性回归课件 page 17, 实际使用在 page 24):

```
# X_train, X_test, y_train, y_test =
train_test_split(X_scaled, y, test_size=0.2,
random_state=42)
```

9. `from sklearn.preprocessing import StandardScaler` (导入类)
    - 用法: 导入用于特征标准化的类。
    - 示例 (来自线性回归课件 page 19):

```
from sklearn.preprocessing import StandardScaler
```

10. `StandardScaler()` (类实例化)
    - 用法: 创建 `StandardScaler` 对象。
    - 示例 (来自线性回归课件 page 19, 实际使用在 page 23):

```
# scaler = StandardScaler()
```

11. `scaler.fit_transform()` (StandardScaler 方法)
    - 用法: `scaler.fit_transform(X)` - 在数据 `X` 上拟合转换器，然后转换它。
    - 示例 (来自线性回归课件 page 19, 实际使用在 page 23):

```
# X_scaled = scaler.fit_transform(X_df)
```

12. `from sklearn.metrics import mean_squared_error, r2_score, mean_absolute_error` (导入函数)
    - 用法: 导入用于模型评估的指标函数。
    - 示例 (来自线性回归课件 page 25, page 28):

```
from sklearn.metrics import mean_squared_error,
r2_score # , mean_absolute_error
```

13. `mean_squared_error()`
    - 用法: `mean_squared_error(y_true, y_pred, *, sample_weight=None, multioutput='uniform_average', squared=True)` - 计算均方误差。
    - 示例 (来自线性回归课件 page 25, page 28):

```
# mse = mean_squared_error(y_test, y_pred)
# print(f'Mean Squared Error: {mse}')
```

14. `r2_score()`
    - 用法: `r2_score(y_true, y_pred, *, sample_weight=None, multioutput='uniform_average')` - R^2 (决定系数) 回归得分函数。
    - 示例 (来自线性回归课件 page 25, page 31):

```
# r2 = r2_score(y_test, y_pred)
# print(f'R^2 Score: {r2}')
```

15. `mean_absolute_error()`
    - 用法: `mean_absolute_error(y_true, y_pred, *, sample_weight=None, multioutput='uniform_average')` - 计算平均绝对误差。
    - 示例 (来自线性回归课件 page 30):

```
# mae = mean_absolute_error(y_test, y_pred)
# print(f'Mean Absolute Error: {mae}')
```

16. `from sklearn.model_selection import cross_val_score` (导入函数)
    - 用法: 导入交叉验证评分函数。
    - 示例 (来自线性回归课件 page 43):

```
from sklearn.model_selection import
cross_val_score
```

17. `cross_val_score()`

- 用法: `cross_val_score(estimator, X, y=None, *, groups=None, scoring=None, cv=None, n_jobs=None, verbose=0, fit_params=None, pre_dispatch='2*n_jobs', error_score=nan)` - 通过交叉验证评估得分。
- 示例 (来自线性回归课件 page 43):

```
# scores = cross_val_score(model, X_scaled, y, cv=5, scoring='r2')
# print(f"Cross-validation scores: {scores}")
# print(f"Average score: {scores.mean()}")
```

18. `from sklearn.model_selection import learning_curve` (导入函数)
    - 用法: 导入学习曲线函数。
    - 示例 (来自线性回归课件 page 45):

```
from sklearn.model_selection import learning_curve
```

19. `learning_curve()`
    - 用法: `learning_curve(estimator, X, y, *, groups=None, train_sizes=array([0.1, 0.33, 0.55, 0.78, 1. ]), cv=None, scoring=None, exploit_incremental_learning=False, n_jobs=None, pre_dispatch='all', verbose=0, shuffle=False, random_state=None, error_score=nan, return_times=False, fit_params=None)` - 确定交叉验证的训练和测试得分，用于不同数量的训练样本。
    - 示例 (来自线性回归课件 page 45):

```
# train_sizes, train_scores, test_scores = learning_curve(
#     model, X, y, cv=5, scoring='neg_mean_squared_error',
#     n_jobs=-1, train_sizes=np.linspace(0.1, 1.0, 5))
```

20. `from sklearn.model_selection import validation_curve` (导入函数)
    - 用法: 导入验证曲线函数。
    - 示例 (来自线性回归课件 page 47):

```
from sklearn.model_selection import validation_curve
```

21. `validation_curve()`

- 用法: `validation_curve(estimator, X, y, *, param_name, param_range, groups=None, cv=None, scoring=None, n_jobs=None, pre_dispatch='all', verbose=0, error_score=nan, fit_params=None)` - 确定交叉验证的训练和测试得分，用于参数的不同值。
- 示例 (来自线性回归课件 page 47):

```
# param_range = np.logspace(-6, -1, 5)
# train_scores, test_scores = validation_curve(
#     SVR(), X, y, param_name="gamma", param_range=param_range,
#     cv=5, scoring="r2", n_jobs=-1)
```

22. `from sklearn.preprocessing import LabelEncoder` (导入类 - 聚类课件)

- 用法: 导入用于将类别标签编码为数字的类。
- 示例 (来自聚类课件 page 31):

```
from sklearn.preprocessing import LabelEncoder
```

23. `LabelEncoder()` (类实例化 - 聚类课件)

- 用法: 创建 `LabelEncoder` 对象。
- 示例 (来自聚类课件 page 31):

```
le = LabelEncoder()
```

24. `le.fit_transform()` (LabelEncoder 方法 - 聚类课件)

- 用法: `le.fit_transform(y)` - 拟合标签编码器然后转换标签为编码标签。
- 示例 (来自聚类课件 page 31):

```
data = ['paris', 'paris', 'tokyo', 'amsterdam']
encoded_data = le.fit_transform(data)
print(encoded_data) # 输出: [1 1 2 0] (或类似，取决于字典序)
```

25. `from sklearn.preprocessing import MinMaxScaler` (导入类 - 聚类课件)

- 用法: 导入用于特征缩放到给定范围的类。
- 示例 (来自聚类课件 page 34):

```
from sklearn.preprocessing import MinMaxScaler
```

26. `MinMaxScaler()` (类实例化 - 聚类课件)
    - 用法: 创建 `MinMaxScaler` 对象。
    - 示例 (来自聚类课件 page 34):

```
scaler = MinMaxScaler(feature_range=(0, 1)) # 缩放到 0-1 范围
```

27. `scaler.fit_transform()` (MinMaxScaler 方法 - 聚类课件)
    - 用法: `scaler.fit_transform(X)` - 在数据 `X` 上拟合转换器，然后转换它。
    - 示例 (来自聚类课件 page 34):

```
data = np.array([[4, 2, 3], [1, 5, 6]])
scaled_data = scaler.fit_transform(data)
print(scaled_data)
```

28. `from sklearn.cluster import KMeans` (导入类 - 聚类课件)
    - 用法: 导入 K-Means 聚类算法类。
    - 示例 (来自聚类课件 page 36):

```
from sklearn.cluster import KMeans
```

29. `KMeans()` (类实例化 - 聚类课件)
    - 用法: `KMeans(n_clusters=8, *, init='k-means++', n_init=10, max_iter=300, tol=0.0001, verbose=0, random_state=None, copy_x=True, algorithm='lloyd')` - 创建 K-Means 聚类模型对象。
    - 示例 (来自聚类课件 page 36):

```
kmeans = KMeans(n_clusters=2, random_state=0)
```

30. `kmeans.fit()` (KMeans 方法 - 聚类课件)
    - 用法: `kmeans.fit(X, y=None, sample_weight=None)` - 计算 K-Means 聚类。

- 示例 (来自聚类课件 page 36):

```
# 假设 X 是准备好的特征数据
# kmeans.fit(X)
```

31. `kmeans.cluster_centers_` (KMeans 属性 - 聚类课件)
    - 用法: 返回聚类中心的坐标。
    - 示例 (来自聚类课件 page 36, page 44):

```
# print(kmeans.cluster_centers_)
```

32. `kmeans.inertia_` (KMeans 属性 - 聚类课件)
    - 用法: 返回样本到其最近聚类中心的平方距离之和（SSE）。
    - 示例 (来自聚类课件 page 36, page 44):

```
# print(kmeans.inertia_)
```

33. `from sklearn.metrics import silhouette_score` (导入函数 - 聚类课件)
    - 用法: 导入计算轮廓系数的函数。
    - 示例 (来自聚类课件 page 46):

```
from sklearn.metrics import silhouette_score
```

34. `silhouette_score()` (聚类课件)
    - 用法: `silhouette_score(X, labels, *, metric='euclidean', sample_size=None, random_state=None, **kwds)` - 计算所有样本的平均轮廓系数。
    - 示例 (来自聚类课件 page 46):

```
# 假设 df 是数据, kmeans.labels_ 是聚类结果标签
# score = silhouette_score(df, kmeans.labels_)
# print(score)
```

35. `kmeans.labels_` (KMeans 属性 - 聚类课件)
    - 用法: 返回每个数据点所属的聚类标签。
    - 示例 (来自聚类课件 page 46, 用于轮廓系数计算):

```
# silhouette_scores.append(silhouette_score(df, kmeans.labels_))
```

Python 与大数据分析 --面向对象思想引入，类的封装调用 (File 3)

(这部分主要是面向对象编程概念，类和方法的定义已在 Python 基础部分提及，这里不再重复。重点是概念和特定方法如 `@classmethod`, `@staticmethod`)

1. `class` (关键字)
    - 用法: `class ClassName(SuperClass): ...` - 用于定义一个类。
    - 示例 (来自课件 page 5):

```python
class Animal:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

2. `__init__()` (特殊方法 - 构造函数)
    - 用法: `def __init__(self, parameters): ...` - 类实例化时自动调用的方法，用于初始化对象的属性。
    - 示例 (来自课件 page 5):

```python
class Animal:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

3. `@classmethod` (装饰器)
    - 用法: `@classmethod def method_name(cls, parameters): ...` - 将一个方法标记为类方法。类方法的第一个参数是类本身（通常命名为 `cls`），而不是实例（`self`）。
    - 示例 (来自课件 page 17):

```python
class Dog:
    num_of_dogs = 0
    def __init__(self, name):
        self.name = name
        Dog.num_of_dogs += 1
    @classmethod
    def get_num_of_dogs(cls):
        return cls.num_of_dogs
print(Dog.get_num_of_dogs())
```

4. `@staticmethod` (装饰器)

- 用法: `@staticmethod def method_name(parameters): ...` - 将一个方法标记为静态方法。静态方法不接收隐式的第一个参数（既不是 `self` 也不是 `cls`）。
- 示例 (来自课件 page 18):

```python
class Math:
    @staticmethod
    def add(x, y):
        return x + y
print(Math.add(5, 10)) # 输出: 15
```

5. `super()` (内置函数)

- 用法: `super().method_name(args)` 或 `super(SubClass, instance_or_class).method_name(args)` - 调用父类（超类）的方法。
- 示例 (来自课件 page 30):

```python
class Rectangle:
    def __init__(self, length, width):
        self.length = length
        self.width = width
class Square(Rectangle):
    def __init__(self, side_length):
        super().__init__(side_length, side_length)
# 调用父类的 __init__
```

6. `__str__()` (特殊方法)

- 用法: `def __str__(self): ...` - 当使用 `print()` 函数或 `str()` 转换对象时调用，返回对象的字符串表示。
- 示例 (来自课件 page 37):

```python
class Car:
    def __init__(self, make, model, year):
        self.make = make
        # ...
    def __str__(self):
        return f"汽车信息：\n 品牌：{self.make}\n 型号：{self.model}\n 年份：{self.year}"
my_car = Car("Toyota", "Camry", 2025)
print(my_car)
```

Python 与大数据分析 --模块和包的简单运用 (File 4 - 实际在 File 3 的 OCR 中)

1. `import module_name` (语句)
    - 用法: 导入指定的模块。
    - 示例 (来自课件 page 45):

```python
# 假设 my_module.py 文件中定义了 say_hello()
import my_module
my_module.say_hello()
```

2. `from module_name import item_name` (语句)
    - 用法: 从模块中导入指定的项（函数、类、变量）。
    - 示例 (来自课件 page 46):

```python
from math import sqrt
print(sqrt(16)) # 直接使用 sqrt，无需 math.sqrt
```

3. `import module_name as alias_name` (语句)
    - 用法: 导入模块并为其指定一个别名。
    - 示例 (来自课件 page 46):

```python
import math as m
print(m.sqrt(16))
```

4. `from . import module_name` (相对导入 - 包内)
    - 用法: `from .sibling_module import item` 或 `from ..parent_package import item` - 在包内部进行相对导入。`.` 表示当前目录，`..` 表示上级目录。

- 示例 (来自课件 page 49):

```python
# 在 my_package/subpackage/submodule1.py 中:
from . import module1 # 假设 module1.py 与 submodule1.py 在同一个包 (subpackage) 内
# 或者如果 module1.py 在 my_package 根目录下:
# from .. import module1
```

collated by zjn , powered by Gemini in 2025