Python上机考试核心内容演示

说明：本Notebook对应考试核心知识点，我将逐单元格运行+讲解，大家可以随时提问~

```
# 演示：变量定义+数据类型查看
num_int = 10  # int类型（整数）
num_float  = 10.0

is_true = True  # bool类型（布尔值：True/False）

# 查看数据类型
print("num_int类型：", type(num_int))
print("num_float类型：", type(num_float))

print("is_true类型：", type(is_true))
```

```
num_int类型： <class 'int'>
num_float类型： <class 'float'>
is_true类型： <class 'bool'>
```

```
# 演示：算术/关系/逻辑运算符
a = 10
b = 3

# 1. 算术运算（加减乘除、取整、取余）
print("加法：", a + b)
print("除法（浮点数）：", a / b)  # 结果是float
print("取整除法：", a // b)  # 只保留整数部分
print("取余（模运算）：", a % b)  # 余数（如10%3=1）

# 2. 关系运算（判断大小，结果是bool）
print("a > b？", a > b)  # 输出True
print("a == b？", a == b)  # 输出False

# 3. 逻辑运算（and/or/not，结合关系运算用）
print("a>5 且 b<5？", (a>5) and (b<5))  # True and True → True
print("a<3 或 b>2？", (a<3) or (b>2))  # False or True → True
```

```
加法： 13
除法（浮点数）： 3.3333333333333335
取整除法： 3
取余（模运算）： 1
a > b？ True
a == b？ False
a>5 且 b<5？ True
a<3 或 b>2？ True
```

```python
# 演示1：分支结构（if-elif-else，按条件执行）
score = 85  # 模拟考试分数
if score >= 90:
    print("优秀")
elif score >= 80:
    print("良好")  # 运行后输出这行
elif score >= 60:
    print("及格")
else:
    print("继续努力")

# 演示2：循环结构（for+while，重复执行）
print("\n===== for循环（遍历容器/范围）=====")
# 遍历列表
for num in [1, 2, 3, 4]:
    print(num, end=" ")  # 输出：1 2 3 4

# 遍历range（考试常用，生成连续整数）
print("\n遍历range(5)：", end=" ")
for i in range(5):
    print(i, end=" ")  # 输出：0 1 2 3 4

print("\n===== while循环（按条件重复）=====")
count = 0
while count < 3:
    print(f"计数：{count}")  # 输出3次：0、1、2
    count += 1  # 计数自增（避免死循环！）
```

```
良好

===== for循环（遍历容器/范围）=====
1 2 3 4
遍历range(5)： 0 1 2 3 4
===== while循环（按条件重复）=====
计数：0
计数：1
计数：2
```

函数及其调用

```python
# 演示：函数定义+参数+返回值（考试高频大题）
def calculate_sum(a, b=0):  # a是必选参数，b是默认参数（可省略）
    """函数说明文档：计算两个数的和（默认第二个数为0）"""
    return a + b  # 返回计算结果

# 函数调用（两种方式）
result1 = calculate_sum(5)  # 只传a，b用默认值0 → 5+0=5
result2 = calculate_sum(5, 3)  # 传a和b → 5+3=8

print("调用1（传1个参数）：", result1)
```

```
print("调用2（传2个参数）：", result2)

# 考试考点：函数嵌套/参数传递/返回值判断
def add_and_double(x, y):
    return calculate_sum(x, y) * 2  # 嵌套调用上面的函数

print("先求和再翻倍：", add_and_double(2, 3))  # (2+3)*2=10
```

```
调用1（传1个参数）： 5
调用2（传2个参数）： 8
先求和再翻倍： 10
```

几种常用且好用结构

```
# 3.1 字符串str（不可变序列，考试重点）
s = "hello python"
print("原字符串：", s)
print("字符串切片（取前5个字符）：", s[0:5])  # 索引0-4 → "hello"
print("字符串切片（倒序取3个）：", s[-3:])  # 倒数3个 → "hon"
print("字符串分割（按空格分）：", s.split(" "))  # 变成列表 → ["hello", "python"]
print("字符串替换：", s.replace("python", "exam"))  # 替换字符

# 3.2 列表list（可变序列，增删改查必考）
lst = [1, 2, 3, 4]
print("\n原列表：", lst)
lst.append(5)  # 末尾新增元素 → [1,2,3,4,5]
print("append后：", lst)
lst.pop(2)  # 删除索引2的元素（3）→ [1,2,4,5]
print("pop(2)后：", lst)
lst[0] = 10  # 修改索引0的元素 → [10,2,4,5]
print("修改后：", lst)
print("列表切片（取中间2个）：", lst[1:3])  # 索引1-2 → [2,4]
```

```
原字符串： hello python
字符串切片（取前5个字符）： hello
字符串切片（倒序取3个）： hon
字符串分割（按空格分）： ['hello', 'python']
字符串替换： hello exam

原列表： [1, 2, 3, 4]
append后： [1, 2, 3, 4, 5]
pop(2)后： [1, 2, 4, 5]
修改后： [10, 2, 4, 5]
列表切片（取中间2个）： [2, 4]
```

```
# 3.3 字典dict（键值对，无序，查询快）
student = {"name": "小明", "age": 18, "score": 85}
print("原字典：", student)
print("通过键取值（考试重点）：", student["name"])  # 键必须存在，否则报错
```

```python
print("get方法取值（推荐）：", student.get("gender", "未知"))  # 键不存在返回默认值

student["gender"] = "男"  # 新增键值对
student["score"] = 90  # 修改值
print("新增/修改后：", student)
print("字典的键：", student.keys())  # 所有键 → dict_keys(['name','age','score','gender'])

# 3.4 集合set（无序、无重复，去重/交集/并集）
s1 = {1, 2, 2, 3, 3}  # 自动去重 → {1,2,3}
s2 = {3, 4, 5}
print("\n集合去重后s1：", s1)
print("集合交集（共有的元素）：", s1 & s2)  # {3}
print("集合并集（所有元素）：", s1 | s2)  # {1,2,3,4,5}
print("集合差集（s1有s2没有）：", s1 - s2)  # {1,2}
```

```
原字典： {'name': '小明', 'age': 18, 'score': 85}
通过键取值（考试重点）： 小明
get方法取值（推荐）： 未知
新增/修改后： {'name': '小明', 'age': 18, 'score': 90, 'gender': '男'}
字典的键： dict_keys(['name', 'age', 'score', 'gender'])

集合去重后s1： {1, 2, 3}
集合交集（共有的元素）： {3}
集合并集（所有元素）： {1, 2, 3, 4, 5}
集合差集（s1有s2没有）： {1, 2}
```

```python
# 演示：冒泡排序（算法基础，考试常考排序题）
def bubble_sort(arr):
    """冒泡排序：相邻元素比较，大的往后移"""
    n = len(arr)
    for i in range(n):
        # 每一轮排好一个最大值，后面的不用比了
        for j in range(0, n - i - 1):
            if arr[j] > arr[j+1]:
                # 交换两个元素（Python特有简洁写法）
                arr[j], arr[j+1] = arr[j+1], arr[j]
    return arr

# 测试排序
test_arr = [5, 2, 9, 1, 3]
print("排序前：", test_arr)
sorted_arr = bubble_sort(test_arr)
print("排序后（升序）：", sorted_arr)

# 考试拓展：求列表最大值（不用max函数）
def find_max(arr):
    max_val = arr[0]
    for num in arr:
        if num > max_val:
            max_val = num
```

```
    return max_val

print("列表最大值：", find_max([2,5,1,8]))  # 8
```

```
排序前： [5, 2, 9, 1, 3]
排序后（升序）： [1, 2, 3, 5, 9]
列表最大值： 8
```

```
# ==================== 一、基础排序算法（考试必考）====================
# 1. 冒泡排序（相邻元素对比交换，稳定排序）
def bubble_sort(arr):
    """
    思路：每一轮遍历将最大元素"冒泡"到末尾，重复n轮（n为数组长度）
    考点：双重循环、元素交换、边界控制
    """
    n = len(arr)
    for i in range(n):
        # 标记是否发生交换（优化：无交换则数组已有序，直接退出）
        swapped = False
        # 每轮后末尾i个元素已排好，无需再比
        for j in range(0, n - i - 1):
            if arr[j] > arr[j+1]:
                arr[j], arr[j+1] = arr[j+1], arr[j]  # 交换
                swapped = True
        if not swapped:
            break
    return arr

# 2. 选择排序（每次选最小元素放到前面，不稳定排序）
def selection_sort(arr):
    """
    思路：遍历数组，找到最小元素的索引，与当前位置交换
    考点：索引查找、元素交换、双重循环优化
    """
    n = len(arr)
    for i in range(n):
        min_idx = i  # 假设当前位置是最小值
        # 查找剩余元素中的最小值索引
        for j in range(i + 1, n):
            if arr[j] < arr[min_idx]:
                min_idx = j
        # 交换当前位置和最小值位置的元素
        arr[i], arr[min_idx] = arr[min_idx], arr[i]
    return arr

# 3. 插入排序（逐个插入到已排序部分，稳定排序）
def insertion_sort(arr):
    """
    思路：将数组分为"已排序区"和"未排序区"，逐个将未排序元素插入对应位置
    考点：元素移位、插入逻辑、单循环+内循环
```

```
    """
    n = len(arr)
    for i in range(1, n):
        key = arr[i]  # 待插入的元素
        j = i - 1  # 已排序区的最后一个索引
        # 已排序区元素比key大则后移
        while j >= 0 and key < arr[j]:
            arr[j + 1] = arr[j]
            j -= 1
        arr[j + 1] = key  # 插入key到正确位置
    return arr

# 测试排序算法
test_arr = [5, 2, 9, 1, 3, 6]
print("=== 排序算法测试 ===")
print("原数组：", test_arr.copy())
print("冒泡排序结果：", bubble_sort(test_arr.copy()))
print("选择排序结果：", selection_sort(test_arr.copy()))
print("插入排序结果：", insertion_sort(test_arr.copy()))


# ==================== 二、查找算法 ====================
# 4. 二分查找（折半查找，仅适用于有序数组）
def binary_search(arr, target):
    """
    思路：将有序数组折半，对比中间元素与目标值，缩小查找范围
    考点：有序前提、边界控制（left<=right）、折半逻辑
    返回：目标值索引（未找到返回-1）
    """
    left = 0
    right = len(arr) - 1  # 右边界为数组最后一个索引

    while left <= right:
        mid = (left + right) // 2  # 中间索引（避免溢出）
        if arr[mid] == target:
            return mid  # 找到目标，返回索引
        elif arr[mid] < target:
            left = mid + 1  # 目标在右半区，左边界右移
        else:
            right = mid - 1  # 目标在左半区，右边界左移
    return -1  # 未找到

# 测试二分查找
sorted_arr = [1, 2, 3, 5, 6, 9]
target1 = 5
target2 = 4
print("\n=== 二分查找测试 ===")
print(f"有序数组：{sorted_arr}")
print(f"查找{target1}，索引：{binary_search(sorted_arr, target1)}")  # 输出3
print(f"查找{target2}，索引：{binary_search(sorted_arr, target2)}")  # 输出-1


# ==================== 三、递归算法（逻辑思维考点）====================
```

```python
# 5. 斐波那契数列（第n项，递归+迭代两种实现）
def fibonacci_recursive(n):
    """
    思路：递归公式 f(n) = f(n-1) + f(n-2)，终止条件 f(0)=0, f(1)=1
    考点：递归终止条件、递归调用逻辑（注意：递归效率低，适合讲解思路）
    """
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibonacci_recursive(n-1) + fibonacci_recursive(n-2)

def fibonacci_iterative(n):
    """
    思路：迭代实现，用两个变量存储前两项，循环计算（考试推荐，效率高）
    考点：循环累加、变量替换
    """
    if n <= 0:
        return 0
    elif n == 1:
        return 1
    a, b = 0, 1  # a=f(0), b=f(1)
    for _ in range(2, n+1):
        a, b = b, a + b  # 每次更新前两项
    return b

# 测试斐波那契数列
n = 10
print("\n=== 斐波那契数列测试 ===")
print(f"第{n}项（递归）：{fibonacci_recursive(n)}")  # 输出55
print(f"第{n}项（迭代）：{fibonacci_iterative(n)}")  # 输出55


# ==================== 四、数学应用算法（基础应用题）====================
# 6. 最大公约数（GCD，欧几里得算法）
def gcd(a, b):
    """
    思路：欧几里得公式 gcd(a,b) = gcd(b, a%b)，终止条件 b=0（此时a为最大公约数）
    考点：模运算、递归/迭代逻辑
    """
    # 迭代实现（更稳定）
    while b != 0:
        a, b = b, a % b
    return a

# 7. 质数判断（判断一个数是否为质数）
def is_prime(num):
    """
    思路：质数是大于1的整数，只能被1和自身整除；优化：只需判断到sqrt(num)
    考点：边界判断、循环优化、取余运算
    """
    if num <= 1:
```

```
        return False  # 1及以下不是质数
    if num == 2:
        return True  # 2是唯一的偶数质数
    if num % 2 == 0:
        return False  # 偶数（除2外）不是质数
    # 只需判断到sqrt(num)，因为超过sqrt(num)的因数会与前面的成对出现
    for i in range(3, int(num**0.5) + 1, 2):
        if num % i == 0:
            return False
    return True

# 测试数学算法
print("\n=== 数学应用算法测试 ===")
print(f"12和18的最大公约数：{gcd(12, 18)}")  # 输出6
print(f"7是否为质数：{is_prime(7)}")  # 输出True
print(f"15是否为质数：{is_prime(15)}")  # 输出False
print(f"2是否为质数：{is_prime(2)}")  # 输出True
```

```
=== 排序算法测试 ===
原数组： [5, 2, 9, 1, 3, 6]
冒泡排序结果： [1, 2, 3, 5, 6, 9]
选择排序结果： [1, 2, 3, 5, 6, 9]
插入排序结果： [1, 2, 3, 5, 6, 9]

=== 二分查找测试 ===
有序数组：[1, 2, 3, 5, 6, 9]
查找5，索引：3
查找4，索引：-1

=== 斐波那契数列测试 ===
第10项（递归）：55
第10项（迭代）：55

=== 数学应用算法测试 ===
12和18的最大公约数：6
7是否为质数：True
15是否为质数：False
2是否为质数：True
```

面向对象与数据结构初步

```
# 5.1 面向对象（类与对象，基础考点）
class Student:
    # 构造方法：初始化对象属性
    def __init__(self, name, age):
        self.name = name  # 实例属性
        self.age = age

    # 成员方法：对象的行为
    def introduce(self):
        print(f"我是{self.name}，今年{self.age}岁")
```

```python
# 创建对象（实例化）
stu1 = Student("小红", 19)
stu2 = Student("小李", 20)

# 调用对象方法
print("学生1介绍：", end="")
stu1.introduce()
print("学生2介绍：", end="")
stu2.introduce()

# 5.2 数据结构初步（列表模拟栈：先进后出）
class Stack:
    def __init__(self):
        self.stack = []  # 用列表存储栈元素

    def push(self, item):  # 入栈（添加元素到栈顶）
        self.stack.append(item)
        print(f"入栈：{item}，当前栈：{self.stack}")

    def pop(self):  # 出栈（删除并返回栈顶元素）
        if self.stack:
            item = self.stack.pop()
            print(f"出栈：{item}，当前栈：{self.stack}")
            return item
        else:
            print("栈为空，无法出栈！")
            return None

# 测试栈操作
s = Stack()
s.push("A")
s.push("B")
s.push("C")
s.pop()  # 先出C
s.pop()  # 再出B
```