第一部分：来源于 `python基本语法及变量.pdf`

练习1：基本打印

```
# 需求：分别打印 "Data" 和 "whale"，然后在一行内打印 "Data*whale"，最后打印 "Data-*-whale"
print("Data")
print(____) # 应打印 whale
print("Data", end="*")
print("____") # 应打印 whale
print("Data", "whale", sep=____) # 应打印 Data-*-whale
```

答案：

1. `"whale"`
2. `"whale"`
3. `"-*-"`

练习2：格式化字符串输出

```
x = 10
y = 20
# 需求：打印出 "计算结果：x + y = 30 !"
print(f"计算结果：{x} + {y} = ____ !")
```

答案：

1. `{x+y}`

练习3：一行输入多个值

```
# 需求：输入两个由空格隔开的数字，并分别赋值给 a 和 b
# 例如输入：10 20
a, b = input("请输入两个数字，用空格隔开：").____()
print(f"a的值是：{a}，b的值是：{b}")
```

答案：

1. `split`

练习4：**if-elif-else** 结构

```
# 需求：根据分数进行评级
# 90分及以上为 'A'
# 80分及以上为 'B'
# 70分及以上为 'C'
# 60分及以上为 'D'
# 低于60分为 'F'
def get_grade(score):
    grade = ''
    if score >= 90:
        grade = 'A'
    ____ score >= 80: # 判断是否大于等于80
        grade = 'B'
    elif score >= 70:
        grade = 'C'
    elif score >= ____: # 判断是否大于等于60
        grade = 'D'
    ____:               # 其他情况
        grade = 'F'
    return grade

print(f"85分的等级是: {get_grade(85)}")
print(f"55分的等级是: {get_grade(55)}")
```

答案：

1. `elif`
2. `60`
3. `else`

练习5：**while** 循环

```python
# 需求：使用while循环，计算1到5的累加和
count = 1
total_sum = 0
while ____ <= 5: # 循环条件
    total_sum = total_sum + count
    count = ____ + 1 # 计数器增加
print(f"1到5的累加和是: {total_sum}")
```

答案：

1. `count`
2. `count`

练习6：for 循环与 range

```python
# 需求：使用for循环打印0到4的数字
print("0到4的数字:")
for i in ____(5): # 生成0到4的序列
    print(i)
```

答案：

1. `range`

练习7：for 循环遍历列表

```python
# 需求：遍历水果列表并打印每个水果
fruits = ["苹果", "香蕉", "樱桃"]
print("水果列表:")
for fruit ____ fruits: # 遍历关键字
    print(fruit)
```

答案：

1. `in`

练习8：嵌套循环（九九乘法表）

```python
# 需求：打印九九乘法表
for i in range(1, 10):
    for j in range(1, ____ + 1): # 内层循环的范围
        print(f'{j}x{i}={i*j}', end='\t')
    ____() # 每打印完一行后换行
```

答案：

1. `i`
2. `print`

练习9：**break** 语句

```python
# 需求：在数字列表中找到第一个大于5的数字后停止查找
numbers = [1, 3, 7, 4, 9, 2]
found_number = None
for num in numbers:
    if num > 5:
        found_number = num
        ____ # 找到后跳出循环
print(f"找到的第一个大于5的数字是: {found_number}")
```

答案：

1. `break`

练习10：**continue** 语句

```python
# 需求：打印1到10之间所有的奇数
print("1到10之间的奇数:")
for i in range(1, 11):
    if i % 2 == 0: # 如果是偶数
        ____       # 跳过当前迭代
    print(i)
```

答案：

1. `continue`

练习11：函数定义与调用

```
# 需求：定义一个函数，计算两个数字的和，并调用它
____ add_numbers(num1, num2): # 定义函数关键字
    result = num1 + num2
    ____ result           # 返回结果关键字

sum_val = add_numbers(5, 3)
print(f"5和3的和是: {sum_val}")
```

答案：

1. `def`
2. `return`

第二部分：来源于 `python函数的使用.pdf`

练习12：函数参数与作用域

```
message = "全局消息"

def greet(name):
    message = "局部消息" # 此处定义的是局部变量
    print(f"{____}, {name}!") # 应该使用哪个message?

greet("张三")
print(f"函数外部的消息: {message}")
```

答案：

1. `message` (函数会优先使用局部作用域的 `message`)

练习13：**global** 关键字

```
count = 0

def increment_counter():
    ____ count # 声明要修改全局变量
    count += 1

increment_counter()
increment_counter()
print(f"计数器的值: {count}") # 期望输出 2
```

答案：

1. `global`

练习14：lambda 函数

```
# 需求：使用lambda函数计算一个数的平方
square = ____ x: x * x # lambda定义
result = square(5)
print(f"5的平方是: {result}")
```

答案：

1. `lambda`

练习15：map 函数与 lambda

```
# 需求：将列表中的每个数字都乘以2
numbers = [1, 2, 3, 4, 5]
doubled_numbers = list(____(lambda x: x * 2, numbers)) # 使用map和lambda
print(f"原列表: {numbers}")
print(f"乘以2后的列表: {doubled_numbers}")
```

答案：

1. `map`

练习16：filter 函数与 lambda

```python
# 需求：筛选出列表中的所有偶数
numbers = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
even_numbers = list(____(lambda x: x % 2 == 0, numbers)) # 使用filter和lambda
print(f"原列表: {numbers}")
print(f"列表中的偶数: {even_numbers}")
```

答案：

1. `filter`

练习17：**sorted 函数与 lambda (作为key)**

```python
# 需求：根据元组的第二个元素对列表进行排序
data = [('苹果', 5), ('香蕉', 2), ('樱桃', 8)]
sorted_data = sorted(data, key=____ x: x[1]) # 使用lambda作为排序的key
print(f"原始数据: {data}")
print(f"按第二个元素排序后: {sorted_data}")
```

答案：

1. `lambda`

第三部分：来源于 `List，Tuple，Set，和Dict.pdf`

练习18：列表创建和访问

```python
# 创建一个包含数字1到5的列表
my_list = [1, 2, 3, 4, 5]
# 访问列表的第一个元素
first_element = my_list[____] # 第一个元素的索引
# 访问列表的最后一个元素
last_element = my_list[____]  # 最后一个元素的索引
print(f"第一个元素: {first_element}")
print(f"最后一个元素: {last_element}")
```

答案：

1. `0`
2. `-1` (或者 `4`)

练习19：列表切片

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
# 需求：获取索引从2到5（不包括5）的元素
sub_list = numbers[2:____]
print(f"子列表 (索引2到4): {sub_list}")
# 需求：获取从开始到索引3（不包括3）的元素
start_to_3 = numbers[:____]
print(f"从开始到索引2的子列表: {start_to_3}")
# 需求：获取从索引7到末尾的元素
from_7_to_end = numbers[____:]
print(f"从索引7到末尾的子列表: {from_7_to_end}")
```

答案：

1. `5`
2. `3`
3. `7`

练习20：列表方法 - **append** 和 **insert**

```
my_list = [1, 2, 3]
my_list.____(4)  # 在列表末尾添加元素4
print(f"添加4后: {my_list}")
my_list.____(1, 1.5) # 在索引1的位置插入元素1.5
print(f"在索引1插入1.5后: {my_list}")
```

答案：

1. `append`
2. `insert`

练习21：列表方法 - **pop** 和 **remove**

```
my_list = [10, 20, 30, 40, 30]
removed_element = my_list.____(2) # 删除并返回索引为2的元素
print(f"删除索引2的元素({removed_element})后: {my_list}")
my_list.____(30) # 删除第一个值为30的元素
print(f"删除第一个30后: {my_list}")
```

答案：

1. `pop`
2. `remove`

练习22：列表推导式

```
# 需求：创建一个包含0到9的平方数的列表
squares = [x____2 for x in range(10)] # 列表推导式
print(f"0到9的平方数列表: {squares}")

# 需求：从一个列表中筛选出所有偶数，并将其乘以2
original_numbers = [1, 2, 3, 4, 5, 6]
processed_numbers = [x * 2 for x in original_numbers if x % 2 ____ 0] # 带条件的列表推导式
print(f"处理后的偶数列表: {processed_numbers}")
```

答案：

1. `**`
2. `==`

练习23：元组创建和不可变性

```
# 创建一个元组
my_tuple = (1, "hello", 3.14)
print(f"元组的第一个元素: {my_tuple[____]}")

# 尝试修改元组的元素 (这会引发错误，因为元组是不可变的)
# my_tuple[0] = 10 # 这行会报错，请注释掉它来运行其他部分

# 元组解包
a, b, c = ____
print(f"a={a}, b={b}, c={c}")
```

答案：

1. `0`
2. `my_tuple`

练习24：集合创建和基本操作

```
# 从列表创建集合，自动去重
my_set = set([1, 2, 2, 3, 4, 4, 4])
print(f"集合: {my_set}")

# 添加元素到集合
my_set._____(5) # 添加元素的方法
print(f"添加5后: {my_set}")

# 检查元素是否存在
print(f"3是否在集合中: {3 ____ my_set}")
print(f"6是否在集合中: {6 ____ my_set}")
```

答案：

1. `add`
2. `in`
3. `in`

练习25：集合运算

```
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}

# 并集
union_set = set1 | ____
print(f"并集: {union_set}")

# 交集
intersection_set = set1 ____ set2
print(f"交集: {intersection_set}")

# 差集 (set1中有，set2中没有的)
difference_set = set1 - ____
```

```
print(f"差集 (set1 - set2): {difference_set}")
```

答案：

1. `set2`
2. `&`
3. `set2`

练习26：字典创建和访问

```
# 创建一个字典
my_dict = {"name": "张三", "age": 30, "city": "北京"}

# 访问字典中的值
student_name = my_dict["____"] # 访问键为"name"的值
student_age = my_dict.____("age") # 使用get方法访问键为"age"的值
print(f"姓名: {student_name}, 年龄: {student_age}")

# 添加新的键值对
my_dict["occupation"] = "工程师"
print(f"添加职业后: {my_dict}")

# 修改值
my_dict["____"] = 31 # 修改年龄
print(f"修改年龄后: {my_dict}")
```

答案：

1. `name`
2. `get`
3. `age`

练习27：字典遍历

```python
my_dict = {"a": 1, "b": 2, "c": 3}

print("遍历键:")
for key in my_dict.____(): # 遍历键的方法
    print(key)

print("遍历值:")
for value in my_dict.____(): # 遍历值的方法
    print(value)

print("遍历键值对:")
for key, value in my_dict.____(): # 遍历键值对的方法
    print(f"{key}: {value}")
```

答案：

1. `keys`
2. `values`
3. `items`

第四部分：来源于 `NumPy介绍、安装及其操作.pdf`

练习28：NumPy数组创建

```python
import numpy as np

# 从列表创建一维数组
arr1d = np.____([1, 2, 3, 4, 5]) # 创建数组的函数
print(f"一维数组: {arr1d}")

# 创建一个3x3的全零数组
zeros_arr = np.zeros((3, ____)) # 指定形状
print(f"3x3全零数组:\n{zeros_arr}")

# 创建一个2x4的全一数组
ones_arr = np.____((2, 4)) # 创建全一数组的函数
print(f"2x4全一数组:\n{ones_arr}")

# 使用arange创建数组
range_arr = np.arange(0, 10, ____) # 从0到9，步长为2
```

```
print(f"步长为2的数组: {range_arr}")
```

答案：

1. `array`
2. `3`
3. `ones`
4. `2`

练习29：NumPy数组属性

```
import numpy as np

arr = np.array([[1, 2, 3], [4, 5, 6]])
print(f"数组形状: {arr.____}")     # 获取形状的属性
print(f"数组维度: {arr.____}")     # 获取维度的属性
print(f"数组元素总数: {arr.____}") # 获取元素总数的属性
print(f"数组数据类型: {arr.____}") # 获取数据类型的属性
```

答案：

1. `shape`
2. `ndim`
3. `size`
4. `dtype`

练习30：NumPy数组 reshape

```
import numpy as np

arr = np.arange(12) # 0到11的数组
print(f"原始数组: {arr}")

# 将一维数组重塑为3x4的二维数组
reshaped_arr = arr.____((3, 4)) # 重塑方法
print(f"3x4数组:\n{reshaped_arr}")
```

答案：

1. `reshape`

练习31：NumPy数组索引和切片

```python
import numpy as np

arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])

# 获取第二行 (索引为1) 的所有元素
row_1 = arr[____, :]
print(f"第二行: {row_1}")

# 获取第三列 (索引为2) 的所有元素
col_2 = arr[:, ____]
print(f"第三列: {col_2}")

# 获取子数组：前两行，第2到3列
sub_arr = arr[____, 1:3]
print(f"子数组 (前两行, 2-3列):\n{sub_arr}")
```

答案：

1. `1`
2. `2`
3. `:2` (或者 `0:2`)

练习32：NumPy布尔索引

```python
import numpy as np

arr = np.array([10, 20, 5, 15, 25, 8])
# 需求：找出数组中所有大于15的元素
greater_than_15 = arr[arr ____ 15] # 布尔条件
print(f"大于15的元素: {greater_than_15}")
```

答案：

1. `>`

练习33：NumPy数组运算

```
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

# 逐元素加法
sum_arr = a + ____
print(f"a + b = {sum_arr}")

# 逐元素乘法
prod_arr = a * b
print(f"a * b = {prod_arr}")

# 数组与标量运算
scaled_arr = a * ____
print(f"a * 2 = {scaled_arr}")
```

答案：

1. b
2. 2

练习34：NumPy通用函数 (ufunc)

```
import numpy as np

arr = np.array([0, np.pi/2, np.pi])
# 计算数组中每个元素的正弦值
sin_arr = np.____(arr) # 正弦函数
print(f"sin值: {sin_arr}")

# 计算数组中所有元素的和
sum_val = np.____(arr) # 求和函数
print(f"数组元素和: {sum_val}")
```

答案：

1. sin

2. `sum`

练习35：NumPy数组转置和点积

```python
import numpy as np

matrix_a = np.array([[1, 2], [3, 4]])
matrix_b = np.array([[5, 6], [7, 8]])

# 矩阵转置
transposed_a = matrix_a.____ # 转置属性
print(f"矩阵A的转置:\n{transposed_a}")

# 矩阵点积 (矩阵乘法)
dot_product = np.____(matrix_a, matrix_b) # 点积函数
print(f"矩阵A和B的点积:\n{dot_product}")
```

答案：

1. `T`
2. `dot`

第五部分：来源于 `pandas的使用.pdf` 和 `pandas高级操作-数据处理.pdf`

练习36：Pandas Series 创建

```python
import pandas as pd

# 从列表创建Series
data_list = [10, 20, 30, 40]
s1 = pd.____(data_list) # 创建Series的函数
print("从列表创建的Series:")
print(s1)

# 从带有自定义索引的列表创建Series
index_labels = ['a', 'b', 'c', 'd']
s2 = pd.Series(data_list, index=____)
print("\n带自定义索引的Series:")
print(s2)
```

答案：

1. `Series`
2. `index_labels`

练习37：Pandas DataFrame 创建

```
import pandas as pd

# 从字典创建DataFrame，字典的键作为列名
data_dict = {'col1': [1, 2, 3], 'col2': [4, 5, 6]}
df1 = pd.____(data_dict) # 创建DataFrame的函数
print("从字典创建的DataFrame:")
print(df1)
```

答案：

1. `DataFrame`

练习38：Pandas 读取CSV文件

```
import pandas as pd

# 假设当前目录下有一个名为 'my_data.csv' 的文件
# df = pd.____('my_data.csv') # 读取CSV的函数
# print(df.head()) # 打印前5行

# 填空：你需要填入正确的pandas函数名来读取CSV文件
# （由于无法实际运行，此处仅为填空）
read_csv_function = "pd.____('my_data.csv')"
```

答案：

1. `read_csv`

练习39：DataFrame 基本操作 - 选择列

```
import pandas as pd

data = {'A': [1, 2, 3], 'B': [4, 5, 6], 'C': [7, 8, 9]}
```

```
df = pd.DataFrame(data)

# 选择列 'A'
col_A = df['____']
print("列 'A':")
print(col_A)

# 选择多列 'A' 和 'C'
cols_AC = df[[____, 'C']]
print("\n列 'A' 和 'C':")
print(cols_AC)
```

答案：

1. `A`
2. `'A'`

练习40：DataFrame 索引 - loc 和 iloc

```
import pandas as pd

data = {'A': [10, 20, 30, 40], 'B': [50, 60, 70, 80]}
df = pd.DataFrame(data, index=['r1', 'r2', 'r3', 'r4'])

# 使用 loc 按标签选择行 'r2'
row_r2_loc = df.____['r2']
print("使用loc选择行 'r2':")
print(row_r2_loc)

# 使用 iloc 按整数位置选择第二行 (索引为1)
row_1_iloc = df.____[1]
print("\n使用iloc选择第二行:")
print(row_1_iloc)

# 使用 loc 选择行 'r1' 到 'r3'，列 'B'
sub_df_loc = df.loc['r1':'r3', ____]
print("\n使用loc选择子DataFrame:")
print(sub_df_loc)
```

答案：

1. `loc`
2. `iloc`
3. `'B'`

练习41：**DataFrame** 条件筛选

```
import pandas as pd

data = {'name': ['Alice', 'Bob', 'Charlie', 'David'],
        'age': [25, 30, 22, 35],
        'score': [88, 92, 78, 85]}
df = pd.DataFrame(data)

# 筛选出年龄大于25的学生
older_students = df[df['age'] ____ 25]
print("年龄大于25的学生:")
print(older_students)
```

答案：

1. `>`

练习42：**Pandas GroupBy** 操作

```
import pandas as pd

data = {'Category': ['A', 'B', 'A', 'B', 'A'],
        'Value': [10, 15, 20, 25, 30]}
df = pd.DataFrame(data)

# 按 'Category' 分组并计算每组的 'Value' 的平均值
grouped_mean = df.____('Category')['Value'].mean() # 分组和聚合
print("按Category分组的Value平均值:")
print(grouped_mean)
```

答案：

1. `groupby`

练习43：**Pandas** 合并 **(Merge)**

```python
import pandas as pd

df1 = pd.DataFrame({'key': ['A', 'B', 'C'], 'value1': [1, 2, 3]})
df2 = pd.DataFrame({'key': ['B', 'C', 'D'], 'value2': [4, 5, 6]})

# 根据 'key' 列合并 df1 和 df2 (内连接)
merged_df = pd.____(df1, df2, on='key', how='inner') # 合并函数
print("合并后的DataFrame (内连接):")
print(merged_df)
```

答案：

1. `merge`

练习44：Pandas 字符串操作 (.str)

```python
import pandas as pd

s = pd.Series(['apple pie', 'banana bread', 'cherry cake'])

# 将Series中的所有字符串转换为大写
upper_s = s.str.____() # 转换为大写的方法
print("大写字符串:")
print(upper_s)

# 检查每个字符串是否包含 'apple'
contains_apple = s.str.____('apple') # 包含检查方法
print("\n是否包含 'apple':")
print(contains_apple)
```

答案：

1. `upper`
2. `contains`

练习45：Pandas 缺失值处理

```python
import pandas as pd
import numpy as np

```

```
data = {'A': [1, 2, np.nan, 4], 'B': [5, np.nan, np.nan, 8]}
df = pd.DataFrame(data)
print("原始DataFrame:")
print(df)

# 删除包含任何缺失值的行
df_dropped = df.____() # 删除缺失值的方法
print("\n删除缺失值后的DataFrame:")
print(df_dropped)

# 用0填充所有缺失值
df_filled = df.____(0) # 填充缺失值的方法
print("\n用0填充缺失值后的DataFrame:")
print(df_filled)
```

答案：

1. `dropna`
2. `fillna`

第六部分：来源于 `数据可视化基础.pdf`

练习46：Matplotlib 绘制简单折线图

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

x = np.array([1, 2, 3, 4, 5])
y = np.array([2, 4, 1, 3, 5])

plt.____(x, y) # 绘制折线图的函数
plt.xlabel("X轴标签")
plt.ylabel("Y轴标签")
plt.title("简单的折线图")
plt.____() # 显示图形的函数
```

答案：

1. `plot`

2. `show`

练习47：**Matplotlib** 绘制散点图

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

x = np.random.rand(50)
y = np.random.rand(50)
colors = np.random.rand(50)
sizes = 1000 * np.random.rand(50)

plt.____(x, y, c=colors, s=sizes, alpha=0.5) # 绘制散点图的函数
plt.xlabel("X值")
plt.ylabel("Y值")
plt.title("散点图示例")
plt.show()
```

答案：

1. `scatter`

练习48：**Matplotlib** 绘制柱状图

```
import matplotlib.pyplot as plt

categories = ['A', 'B', 'C', 'D']
values = [10, 24, 15, 30]

plt.____(categories, values, color='skyblue') # 绘制柱状图的函数
plt.xlabel("类别")
plt.ylabel("值")
plt.title("柱状图示例")
plt.show()
```

答案：

1. `bar`

练习49：**Matplotlib** 绘制直方图

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

data = np.random.randn(1000) # 生成1000个标准正态分布的随机数

plt.____(data, bins=30, color='green', alpha=0.7) # 绘制直方图的函数
plt.xlabel("值")
plt.ylabel("频率")
plt.title("直方图示例")
plt.show()
```

答案：

1. `hist`

练习50：**Seaborn** 绘制热力图

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

# 创建一个随机的10x10矩阵
data_matrix = np.random.rand(10, 10)

sns.____(data_matrix, annot=True, cmap="YlGnBu") # Seaborn绘制热力图的函数
plt.title("热力图示例")
plt.show()
```

答案：

1. `heatmap`

第八部分：**Python**与大数据分析 -- 数据分析 **(Titanic** 示例**)**
(这里只选与 Pandas 操作直接相关的填空，模型训练部分过于复杂)

24. **Pandas** 数据加载与初步探索 **(Titanic)**

```
import pandas as pd
```

```
2
3  # 假设 titanic_train.csv 文件在当前目录
4  train_df = pd.read_csv('titanic_train.csv') # 此处假设你已
   下载该文件
5
6  print("数据集头部:")
7  print(train_df.____(3)) # 显示前3行
8
9  print("\n缺失值统计:")
10 print(train_df.isnull().____()) # 统计每列的缺失值数量
11
12 print("\n数据集基本信息:")
13 train_df.info()
14
15 print("\n数值特征描述性统计:")
16 print(train_df.____()) # 获取数值列的描述性统计
```

空白处应填写：

- 第一个 `____`: `head`
- 第二个 `____`: `sum`
- 第三个 `____`: `describe`

25. **Pandas 数据清洗与转换 (Titanic)**

```
1  import pandas as pd
2  import numpy as np
3  # 假设 train_df 已经加载
4  train_df = pd.read_csv('titanic_train.csv')
5
6  # 填充 'Age' 列的缺失值，使用年龄的中位数
7  age_median = train_df['Age'].____()
8  train_df['Age'].fillna(age_median, inplace=True)
9
10 # 将 'Sex' 列转换为数值 (male:0, female:1)
11 train_df['Sex_Numeric'] = train_df['Sex'].____({'male':
   0, 'female': 1})
12
13 # 删除不必要的列，例如 'Ticket' 和 'Cabin'
14 columns_to_drop = ['Ticket', 'Cabin', 'Name',
   'PassengerId']
15 train_df_cleaned =
   train_df.____(columns=columns_to_drop, axis=1)
```

```
16
17 print("\n处理后数据集头部:")
18 print(train_df_cleaned.head())
19
20 print("\n处理后缺失值统计:")
21 print(train_df_cleaned.isnull().sum())
```

空白处应填写：

- 第一个 `____`: `median`
- 第二个 `____`: `map`
- 第三个 `____`: `drop`

第九部分：**Python**与大数据分析 -- 线性回归
(主要关注 Sklearn 的使用)

26. **Sklearn** 线性回归基本流程

```
1  from sklearn.model_selection import train_test_split
2  from sklearn.linear_model import ____ # 导入线性回归模型
3  from sklearn.metrics import mean_squared_error
4  import numpy as np
5  import pandas as pd
6
7  # 假设 X_data 是特征DataFrame, y_data 是目标Series
8  # 造一些简单数据为例
9  rng = np.random.RandomState(1)
10 X_data = rng.rand(50, 2)
11 y_data = 2 * X_data[:, 0] - 3 * X_data[:, 1] +
   rng.randn(50) * 0.5
12
13 # 划分训练集和测试集
14 X_train, X_test, y_train, y_test = ____(X_data, y_data,
   test_size=0.2, random_state=42)
15
16 # 创建并训练模型
17 model = LinearRegression()
18 model.____(X_train, y_train) # 训练模型
19
20 # 进行预测
21 y_pred = model.predict(____) # 在测试集上预测
```

```
22
23 # 评估模型
24 mse = mean_squared_error(y_test, y_pred)
25 print(f"均方误差 (MSE): {mse:.2f}")
26 print(f"模型系数 (slopes): {model.coef_}")
27 print(f"模型截距 (intercept): {model.intercept_:.2f}")
```

空白处应填写：

- 第一个 `____`: `LinearRegression`
- 第二个 `____`: `train_test_split`
- 第三个 `____`: `fit`
- 第四个 `____`: `X_test`

第十部分：Python与大数据分析 -- 聚类 (KMeans)

27. Sklearn KMeans聚类

```
1  from sklearn.cluster import ____ # 导入KMeans模型
2  from sklearn.preprocessing import StandardScaler
3  import numpy as np
4  import pandas as pd
5
6  # 假设有特征数据 X_cluster_data
7  # 造一些简单数据为例
8  X_cluster_data = np.array([[1, 2], [1.5, 1.8], [5, 8],
   [8, 8], [1, 0.6], [9, 11]])
9
10 # 数据标准化 (聚类前通常需要)
11 scaler = StandardScaler()
12 X_scaled = scaler.fit_transform(____)
13
14 # 创建KMeans模型，假设分为2个簇
15 kmeans = KMeans(n_clusters=2, random_state=42,
   n_init='auto')
16 kmeans.____(X_scaled) # 训练聚类模型
17
18 # 获取聚类标签和簇中心
19 labels = kmeans.labels_
20 centroids = kmeans.____
21
```

```
print("每个样本的聚类标签:", labels)
print("簇中心:\n", centroids)
```

空白处应填写：

- 第一个 `____`: `KMeans`
- 第二个 `____`: `X_cluster_data`
- 第三个 `____`: `fit`
- 第四个 `____`: `cluster_centers_`

第十一部分：Python与大数据分析 -- 主成分分析 (PCA)

28. Sklearn PCA降维

```
from sklearn.decomposition import ____ # 导入PCA模型
from sklearn.preprocessing import StandardScaler
import numpy as np

# 假设有高维特征数据 X_pca_data
# 造一些简单数据为例 (5个样本, 4个特征)
X_pca_data = np.array([[1,2,3,4], [5,5,6,7], [1,1,2,2],
[8,9,7,8], [4,5,4,6]])

# 数据标准化 (PCA前通常需要)
scaler = StandardScaler()
X_scaled_pca = scaler.fit_transform(X_pca_data)

# 创建PCA模型，降到2维
pca = PCA(n_components=____)
X_pca_transformed = pca.____(X_scaled_pca) # 应用PCA降维

print("原始数据形状:", X_scaled_pca.shape)
print("降维后数据形状:", X_pca_transformed.shape)
print("解释的方差比例:", pca.explained_variance_ratio_)
```

空白处应填写：

- 第一个 `____`: `PCA`
- 第二个 `____`: `2`
- 第三个 `____`: `fit_transform`