Python基础回顾练习

输入输出练习

In [ ]:
```
student1 = '学生1'
student2 = '学生2'
```

1. 分别打印两个变量
2. 在同一行打印两个变量 学生1,学生2
3. 输出 学生1是学生2的朋友

In [ ]:
```
print(student1)
print(student2)
```

In [ ]:
```
print(student1, end=' ')
print(student2)
```

In [ ]:
```
print(student1, student2)
```

In [ ]:
```
print(f"{student1}是{student2}的朋友！")
```

In [ ]:
```
import time
time_str = time.asctime()
print(f"现在是:{time_str}")
```

In [ ]:
```
a = 1
b = 2
print(f"{b} 大于 {a}")
```

字符串的操作练习

In [ ]:
```
a = 'zhipeng\t176\t60\t男\nJames\t178\t65\t女\nabc\t160\t55\t女'
print(a)
```

1. 获取字符串中间的字符
2. 将zhipeng 替换成 叶志鹏
3. 将字符串转大小写
4. 将字符串进行拼接
5. 判断字符串是不是数字字符
6. 删除一段文本的空格与换行符
7. 切割字符串

In [ ]:
```
index = len(a)-1
a[index]
```

In [ ]:
```
print(a[len(a)//2])
```

In [ ]:
```
a[len(a)//2]
```

In [ ]:
```
a[0:len(a)//2]
```

In [ ]:
```
a[len(a)//2:len(a)-1]
```

In [ ]:
```
a[0:2]
```

In [ ]:
```
a.replace("zhipeng", "叶志鹏")
```

In [ ]:
```
a.lower()
```

In [ ]:
```
a.upper()
```

In [ ]:
```
a = "Hello"
b = " World!"

c = a+b
print(c)
d = f"{a}{b}"
print(d)
```

In [ ]:
```
a = "5.1,3.5,1.4,0.2,Iris-setosa"
b = a.split(',')
print(b)
X = b[:4]
Y = b[4]
print(X)
print(Y)
X = [float(value) for value in X]
print(X)
```

分支循环结构练习

1. 写一个计算圆面积的程序，要求必须带有输入圆半径，print 圆的面积。
2. 打印 9*9 乘法表。
3. 判断一个数是不是素数，要求带有输入input，以及 print 结果。
4. 找出 1 ~ n 范围内的所有素数，要求输入 n, 返回所有素数。
5. 将字符串反转，比如 'abcdef' -> 'fedcba' 。

In [ ]:
```
radius = input("type your radius:")
radius = float(radius)
area = radius**2*3.14
print(f"area:{area}")
```

In [ ]:
```
for i in range(50, 100, 2):
    print(i)
```

In [ ]:
```
start = 50
while start<100:
    print(start)
    start = start + 1
```

In [ ]:
```
for i in range(1, 10):
    for j in range(1, i+1):
        print(f"{i}*{j}={i*j}", end='\t')
    print()
```

In [ ]:
```
start_i = 1
while start_i < 10:
    start_j = 1
    while start_j < start_i + 1:
        print(f"{start_i}*{start_j}={start_i*start_j}", end='\t')
        start_j = start_j + 1
    print()
    start_i = start_i + 1
```

In [ ]:
```
for i in range(1, 10):
    for j in range(1, 11-i):
        print(f"{i}*{j}={i*j}", end='\t')
    print()
```

In [ ]:
```
for i in range(1, 10):
    for j in range(1, i):
        print('\t', end='')
    for k in range(1, 11-i):
        print(f"{i}*{k}={i*k}", end='\t')
    print()
```

质数 不能被除1和自身之外的数整除 就是质数

In [ ]:
```
num = int(input("type num:"))
flag = True
for i in range(2, num):
    if num % i == 0:
        flag = False

if flag:
    print(f"{num} 是质数")
else:
    print(f"{num} 不是质数")
```

In [ ]:
```
n = int(input("type n:"))
for num in range(2, n):
    flag = True
    for i in range(2, num):
        if num % i == 0:
            flag = False
    if flag:
        print(f"{num} 是质数")
```

In [ ]:
```
test = "Hello World!"
for i in range(len(test)):
    print(test[i])
```

In [ ]:
```
test = "Hello World!"
for i in range(len(test)):
    print(test[len(test)-i-1])
```

In [ ]:
```
test = "Hello World!"
for i in range(len(test)):
    print(test[0:i+1])
```

集合类的练习

In [ ]:
```
scores = [91,98,99,100,80]
students = ['James','Sihui','Knok','Yida','kevin']
```

In [ ]:
```
result = []
for index in range(len(scores)):
    item = (students[index], scores[index])
    result.append(item)
print(result)
```

In [ ]:
```
data = [('James', 91), ('Sihui', 98), ('Knok', 99), ('Yida', 100), ('kevin', 80)]
```

In [ ]:
```
sum_value = 0
for i in range(len(data)):
    sum_value += data[i][1]
print(sum_value/len(data))
```

In [ ]:
```
data = [('James', 91), ('Sihui', 98), ('Knok', 99), ('Yida', 100), ('kevin', 80)]
data.sort(key=lambda item: item[1], reverse=True)
print(data)
```

1. 遍历scores列表，依次打印各个元素
2. 排序列表
3. 将姓名和成绩组成元组tuple，如 ("James", 91)
4. 将姓名和成绩组成字典dictionary，如 {"James":91, 'Sihui':98,...}
5. 遍历字典 key, value
6. 判断某个元素，在不在一个集合中 （in）
7. 列表，字典，tuple，set的增删改查
8. 有10000个样本的label，如何去重得到整个数据集有多少类别?

In [ ]:
```
result = {"name":"zhipeng", "age": 20, "home":"taizhou"}
print(result['home'])
```

In [ ]:
```
for i in range(len(scores)):
    print(scores[i])
```

In [ ]:
```
list(range(len(scores)))
```

In [ ]:
```
for value in scores:
    print(value)
```

In [ ]:
```
scores = sorted(scores)
print(scores)
```

In [ ]:
```
scores = sorted(scores, reverse=True)
print(scores)
```

In [ ]:
```
scores = [91,98,99,100,80]
scores.sort(reverse=True)
print(scores)
```

In [ ]:
```
scores[0] = 60
print(scores)
```

In [ ]:
```
scores = (100, 99, 98, 91, 80)
scores[0] = 60
```

In [ ]:
```
scores = [91,98,99,100,80]
students = ['James','Sihui','Knok','Yida','kevin']
result = []
for i in range(len(scores)):
    item = {}
    item[students[i]] = scores[i]
    result.append(item)
print(result)
```

In [ ]:
```
scores = [91,98,99,100,80]
students = ['James','Sihui','Knok','Yida','kevin']
result = []
for i in range(len(scores)):
    item = {students[i]:scores[i]}
    result.append(item)
print(result)
```

In [ ]:
```
data = {'James': 91, 'Sihui': 98, 'Knok': 99,'Yida': 100,'kevin': 80}
for key in data:
    print(key, data[key])
```

In [ ]:
```
"James" in data
```

In [ ]:
```
import random
labels = ('bb', 'aa', 'cc')
samples = []
for i in range(1000):
    label = random.choice(labels)
    samples.append(label)
print(samples)
```

In [ ]:
```
count_label = 0
tmp = []
for value in samples:
    if value in tmp:
        continue
    else:
        count_label += 1
        tmp.append(value)
print(count_label)
print(tmp)
```

In [ ]:
```
count_label = 0
tmp = []
for value in samples:
    if not value in tmp:
        count_label += 1
        tmp.append(value)
print(count_label)
print(tmp)
```

In [ ]:
```
tmp = set()
for value in samples:
```

```
    tmp.add(value)
print(tmp)
```

函数的练习

1. 为什么需要函数?
2. 函数的定义
3. 函数的参数传值，顺序参数，命名参数，默认参数
4. 函数的值传递（可变类型，不可变类型）
5. python的内置函数 https://docs.python.org/zh-cn/3/library/functions.html
6. 递归调用，实现 $f(x,n)=x^n, f(n)=n!$函数
7. 编写函数计算斐波拉切数列的第n项，参数是第几项n，返回第n项值
8. 假设你正在爬楼梯。需要 n 阶你才能到达楼顶。每次你可以爬 1 或 2 个台阶。你有多少种不同的方法可以爬到楼顶呢？注意：给定 n 是一个正整数

In [ ]:
```
def f(x, n):
    if n > 1:
        return x*f(x, n-1)
    else:
        return x
print(f(2,10))
```

In [ ]:
```
def f(x, n):
    if n == 1:
        return x
    else:
        return x*f(x, n-1)
print(f(2,10))
```

函数的位置参数传递

In [ ]:
```
def power(x, n):
    value = 1
    for i in range(n):
        value = value*x
    return value
```

In [ ]:
```
power(2, 10)
```

参数的默认值

In [ ]:
```
def power(x, n=1):
    value = 1
    for i in range(n):
        value = value*x
    return value
```

In [ ]:
```
power(10)
```

命名参数传值

In [ ]:
```
power(n=10, x=1)
```

可变参数传值

In [ ]:
```
def sum(numbers):
    value = 0
    for num in numbers:
        value = value + num
    return value
```

In [ ]:
```
sum([1,2,3,4,5,6])
```

In [ ]:
```
a=1
b=2
print(a)
print(a,b)
```

In [ ]:
```
def sum(*numbers):
    value = 0
    for num in numbers:
        value = value + num
    return value
```

In [ ]:
```
sum(1,2,3,4,5,6)
```

In [ ]:
```
def sum(*numbers, **kw):
    value = 0
    for num in numbers:
        value = value + num
    print(kw)
    return value
```

In [ ]:
```
sum(1,2,3,4,5,6, name = 'test', age = 20, home="taizhou")
```

In [ ]:
```
a = 20
def add_one(num):
    num = num + 1
```

In [ ]:
```
add_one(a)
print(a)
```

In [ ]:
```
a = 20
def add_one(num):
    num = num + 1
    return num
```

In [ ]:
```
a = add_one(a)
print(a)
```

In [ ]:
```
seq = [1,2,3]
def add_one(nums):
    for i in range(len(nums)):
        nums[i] = nums[i] + 1
    return nums
seq = add_one(seq)
print(seq)
```

In [ ]:
```
a = [1,2,3,4,1,1,2,1,2,4]
for index, value in enumerate(a):
    print(index, value)
```

In [ ]:
```
scores = [91,98,99,100,80]
students = ['James','Sihui','Knok','Yida','kevin']
for score, student in zip(scores, students):
    print(student, score)
```

Python进阶知识

文件读写

open()函数的几种模式

只读模式

In [ ]:
```
file = open('iris.data', 'r')
content = file.read()
file.close()
print(content)
```

In [ ]:
```
try:
    file = open('iris.data', 'r')
    content = file.read()
except:
    print("产生异常！")
finally:
    file.close()
print(content)
```

因为读写文件容易产生异常，所以，常用try except finally 处理

也可以采用 with语法 来管理 open()函数的上下文，自动关闭输入流

In [ ]:
```
with open('iris.data', 'r') as file:
    content = file.read()
print(content)
```

某些大数据场景，不能一次性将数据读入到内存当中，需要分批次读入。

In [ ]:
```
with open('iris.data', 'r') as file:
    for line in file:
        print(line)
```

统计Iris数据集各列的均值、方差、标准差、最后的label的种类，以及按照第一列给数据集进行排序输出。

读写二进制数据（字节数据）

In [ ]:
```
with open('../lenna512color.bmp', 'rb') as file:
    content = file.read()
```

绝对路径： D:\documents\数据可视化\实验一Python进阶\dir2\lenna512color.bmp 相对路径：dir2/lenna512color.bmp ../lenna512color.bmp

字符串编码与解码问题

字符串编码与解码问题

字符串编码与解码问题

In [ ]:
```
content = "我"
content_gbk = content.encode(encoding='gbk')
print(content_gbk)
content_utf8 = content.encode(encoding='utf8')
print(content_utf8)
```

In [ ]:
```
test = b'\xce\xd2'
content_gbk = test.decode(encoding='gbk')
print(content_gbk)
content_utf8 = test.decode(encoding='utf8')
print(content_utf8)
```

In [ ]:
```
with open('test.html', 'r') as file:
    content = file.read()
print(content)
```

当文本文件中夹杂一些非法编码的字符，我们可以添加 errors='ignore'参数，来忽略错误。

In [ ]:
```
file = open('XXXXXX', 'r', encoding='gbk', errors='ignore')
```

写模式

In [ ]:
```
data = [i for i in range(10)]
with open('test.txt', 'w') as file:
    for value in data:
        file.write(str(value))
```

写二进制数据 wb

In [ ]:
```
data = ["我", "你"]
with open("test.txt", 'w') as file:
    for value in data:
        file.write(value)
```

In [ ]:
```
data = ["我", "你"]
with open('test.txt', 'wb') as file:
    for value in data:
        file.write(str(value).encode(encoding='gbk'))
```

追加写 a / ab

In [ ]:
```
data = ["我", "你"]
with open('test.txt', 'a') as file:
    for value in data:
        file.write(str(value))
```

os模块与文件目录

获取系统属性

In [ ]:
```
import os
print(os.name)
```

如果是posix，说明系统是Linux、Unix或Mac OS X，如果是nt，就是Windows系统。

获取环境变量

In [ ]:
```
print(os.environ)
```

```
environ({'ALLUSERSPROFILE': 'C:\\ProgramData', 'APPDATA': 'C:\\Users\\93964\\AppData
\\Roaming', 'APPLICATION_INSIGHTS_NO_DIAGNOSTIC_CHANNEL': '1', 'CHROME_CRASHPAD_PIPE
_NAME': '\\\\.\\pipe\\crashpad_8068_YEWWPWKNBVPDAGVR', 'CLION': 'D:\\apps\\CLion 202
2.2.1\\bin;', 'COMMONPROGRAMFILES': 'C:\\Program Files\\Common Files', 'COMMONPROGRA
MFILES(X86)': 'C:\\Program Files (x86)\\Common Files', 'COMMONPROGRAMW6432': 'C:\\Pr
ogram Files\\Common Files', 'COMPUTERNAME': 'FOURIERYE', 'COMSPEC': 'C:\\WINDOWS\\sy
stem32\\cmd.exe', 'CONDA_DEFAULT_ENV': 'base', 'CONDA_EXE': 'C:\\ProgramData\\Anacon
da3\\condabin\\..\\Scripts\\conda.exe', 'CONDA_PREFIX': 'C:\\ProgramData\\Anaconda
3', 'CONDA_PROMPT_MODIFIER': '(base) ', 'CONDA_PYTHON_EXE': 'C:\\ProgramData\\Anacon
da3\\python.exe', 'CONDA_SHLVL': '1', 'DRIVERDATA': 'C:\\Windows\\System32\\Drivers
\\DriverData', 'ELECTRON_RUN_AS_NODE': '1', 'HOMEDRIVE': 'C:', 'HOMEPATH': '\\Users
\\93964', 'JPY_INTERRUPT_EVENT': '1936', 'LOCALAPPDATA': 'C:\\Users\\93964\\AppData
\\Local', 'LOGONSERVER': '\\\\FOURIERYE', 'NUMBER_OF_PROCESSORS': '8', 'ONEDRIVE':
'C:\\Users\\93964\\OneDrive', 'ORIGINAL_XDG_CURRENT_DESKTOP': 'undefined', 'OS': 'Wi
ndows_NT', 'PATH': 'c:\\ProgramData\\Anaconda3;C:\\ProgramData\\Anaconda3;C:\\Progra
mData\\Anaconda3\\Library\\mingw-w64\\bin;C:\\ProgramData\\Anaconda3\\Library\\usr
\\bin;C:\\ProgramData\\Anaconda3\\Library\\bin;C:\\ProgramData\\Anaconda3\\Scripts;
C:\\ProgramData\\Anaconda3\\bin;C:\\ProgramData\\Anaconda3\\condabin;D:\\apps\\vmwar
e\\bin;C:\\WINDOWS\\system32;C:\\WINDOWS;C:\\WINDOWS\\System32\\Wbem;C:\\WINDOWS\\Sy
stem32\\WindowsPowerShell\\v1.0;C:\\WINDOWS\\System32\\OpenSSH;D:\\apps\\Git\\cmd;
C:\\ProgramData\\Anaconda3;C:\\ProgramData\\Anaconda3\\Scripts;C:\\ProgramData\\Anac
onda3\\Scripts\\Library\\bin;C:\\Program Files (x86)\\oh-my-posh\\bin;C:\\Users\\939
64\\.local\\opt\\archiver-v3.5.0\\bin\\arc.exe;c:\\users\\93964\\appdata\\roaming\\p
ython\\python39\\Scripts;C:\\Users;D:\\apps\\nasm;D:\\apps\\CLion 2022.2.1\\bin;
C:\\Users\\93964\\AppData\\Local\\Microsoft\\WindowsApps', 'PATHEXT': '.COM;.EXE;.BA
T;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC', 'POSH_THEMES_PATH': 'C:\\Program Files (x
86)\\oh-my-posh\\themes', 'PROCESSOR_ARCHITECTURE': 'AMD64', 'PROCESSOR_IDENTIFIER':
'Intel64 Family 6 Model 142 Stepping 12, GenuineIntel', 'PROCESSOR_LEVEL': '6', 'PRO
CESSOR_REVISION': '8e0c', 'PROGRAMDATA': 'C:\\ProgramData', 'PROGRAMFILES': 'C:\\Pro
gram Files', 'PROGRAMFILES(X86)': 'C:\\Program Files (x86)', 'PROGRAMW6432': 'C:\\Pr
ogram Files', 'PROMPT': '(base) $P$G', 'PSMODULEPATH': 'C:\\Program Files\\WindowsPo
werShell\\Modules;C:\\WINDOWS\\system32\\WindowsPowerShell\\v1.0\\Modules', 'PUBLI
C': 'C:\\Users\\Public', 'PYCHARM': 'D:\\apps\\PyCharm 2022.1.3\\bin;', 'PYTHONIOENC
ODING': 'utf-8', 'PYTHONUNBUFFERED': '1', 'SYSTEMDRIVE': 'C:', 'SYSTEMROOT': 'C:\\WI
NDOWS', 'TEMP': 'C:\\Users\\93964\\AppData\\Local\\Temp', 'TMP': 'C:\\Users\\93964
\\AppData\\Local\\Temp', 'USERDOMAIN': 'FOURIERYE', 'USERDOMAIN_ROAMINGPROFILE': 'FO
URIERYE', 'USERNAME': '93964', 'USERPROFILE': 'C:\\Users\\93964', 'VSCODE_AMD_ENTRYP
OINT': 'vs/workbench/api/node/extensionHostProcess', 'VSCODE_CODE_CACHE_PATH':
'C:\\Users\\93964\\AppData\\Roaming\\Code\\CachedData\\da76f93349a72022ca4670c1b8486
0304616aaa2', 'VSCODE_CWD': 'D:\\apps\\Microsoft VS Code', 'VSCODE_HANDLES_UNCAUGHT_
ERRORS': 'true', 'VSCODE_IPC_HOOK': '\\\\.\\pipe\\5c0488400c348ec383357e746b163fa9-
1.70.0-main-sock', 'VSCODE_NLS_CONFIG': '{"locale":"zh-cn","availableLanguages":
{},"_languagePackSupport":true}', 'VSCODE_PID': '8068', 'WINDIR': 'C:\\WINDOWS', 'ZE
S_ENABLE_SYSMAN': '1', '__COMPAT_LAYER': 'RunAsAdmin Installer', 'PYDEVD_USE_FRAME_E
VAL': 'NO', 'TERM': 'xterm-color', 'CLICOLOR': '1', 'PAGER': 'cat', 'GIT_PAGER': 'ca
t', 'MPLBACKEND': 'module://matplotlib_inline.backend_inline'})
```

文件操作

In [ ]:
```
os.rename('a.txt', 'test.txt')
```

In [ ]:
```
os.remove('test.txt')
```

In [ ]:
```
# 查看当前目录的绝对路径:
os.path.abspath('.')
# 在某个目录下创建一个新目录，首先把新目录的完整路径表示出来:
os.path.join('/Users/michael', 'testdir')
# 然后创建一个目录:
os.mkdir('./test')
# 删掉一个目录:
os.rmdir('./test')
# 获取文件夹下的所有文件包括目录
os.listdir()
```

面向对象 （万物皆对象的设计理念）

定义类

In [ ]:
```
class Student():
    def __init__(self, name, score):
        self.name = name
        self.score = score

    def print_score(self):
        print(f"{self.name}'s score is {self.score}")
```

创建对象

In [ ]:
```
student1 = Student("zhipeng", 100)
student2 = Student("test", 80)
print(student1.name)
print(student1.score)
student1.print_score()
```

__私有属性与方法

In [ ]:
```
class Student(object):
    def __init__(self, name, score):
        self.__name = name
        self.__score = score

    def print_score(self):
        print(f"{self.__name}'s score is {self.__score}")

    def get_grade(self):
        if self.__score >=90:
            return 'A'
        elif self.__score >=80:
            return 'B'
        elif self.__score >=60:
            return 'C'
        else:
            return 'D'
```

封装

即隐藏对象的属性和实现细节，仅对外公开接口，控制在程序中属性的读和修改的访问级别；将抽象得到的数据和行为（或功能）相结合，形成一个有机的整体

In [ ]:
```
class Student(object):
    def __init__(self, name, score):
        self.__name = name
        self.__score = score

    def print_score(self):
        print(f"{self.__name}'s score is {self.__score}")

    def get_grade(self):
        if self.__score >=90:
            return 'A'
        elif self.__score >=80:
            return 'B'
        elif self.__score >=60:
            return 'C'
        else:
            return 'D'
    def set_score(self,score):
        if score >=0 and score <= 100:
            self.__score = score
        else:
            print('The score must be in [0, 100]. Please type again!')

    def get_score(self):
        return self.__score
```

继承

In [ ]:
```
class Animal():
    def __init__(self, weight, height):
        self.weight = weight
        self.height = height

    def moving(self):
        print('The animal is moving!')

class Dog(Animal):
    def __init__(self,weight, height, master):
        super(Dog,self).__init__(weight, height)
        self.master = master
    def moving(self):
        print('The dog is running!')
    def watchdoor(self):
        print(f'Hello {self.master} master, I am watching door!')
```

类的常用函数 type()、dir()

类和对象的特殊方法

__str__、 __repr__、__len__、__eq__

利用特殊方法，实现数学操作

万物皆对象

In [ ]:
```
def test():
    print("hello World!")
print(type(test))
```

In [ ]:
```
def f(x):
    return x**2

def fx(fun, x):
    theta = 0.0001
    return (fun(x+theta)-fun(x))/theta
```

In [ ]:
```
def g(x):
    return x**3+x**2+1

print(fx(g, 10))
```

高级特性

迭代

In [ ]:
```
# for value in xxx:
```

列表生成式

In [ ]:
```
a = [x**2 for x in range(10)]
print(a)
```

In [ ]:
```
b = [x**2 for x in range(10) if x**2%2==0]
print(b)
```

In [ ]:
```
c = [x**2 if x**2%2==0 else 0 for x in range(10)]
print(c)
```

生成器

当生成的数据很大的时候，防止内存溢出，常采用生成器。

In [ ]:
```
a = (x**2 for x in range(10))
print(a)
```

In [ ]:
```
next(a)
```

In [ ]:
```
def gen_fx(MAX):
    n = 0
    a, b = 0, 1
    while n <= MAX:
        a, b = b, a+b
        yield a
        n = n + 1
```

In [ ]:
```
f = gen_fx(10)
```

In [ ]:
```
for value in f:
    print(value)
```

In [ ]:
```
range(10)
```

可迭代对象

In [ ]:
```
class Books(object):
    def __init__(self,names):
        self.books = names
        self.index = -1
    def __iter__(self):
        return self
    def __next__(self):
        if self.index >= len(self.books) - 1:
            raise StopIteration
        self.index = self.index + 1
        return self.books[self.index]
```

In [ ]:
```
books = Books(["cs", "EE", "math"])
for value in books:
    print(value)
```

装饰器

In [ ]:
```
@log
def processing(data):
    for index, value in enumerate(data):
        data[index] = value + 1
    return data
```

In [ ]:
```
processing([1,2,3])
```

In [ ]:
```
import time
def log(func):
    def wrapper(*args, **kw):
        date = time.strftime('%Y-%m-%d %H:%M:%S', time.localtime())
        print(date, end='\t')
        return func(*args, **kw)
    return wrapper
```

In [ ]:
```
print(processing([1,2,3]))
```

模块、包的使用