No.

Date

Python 林粒粒呀 2025.10.28-11.1

* Programming之前：①安装 python 解释器

②安装代码编辑器 e.g. Pycharm (IDE：集成开发环境)

* .venv：不要动，不要删

P6 print.

1. print("Dad!!")
   - 单双引号都ok
   - *英文括号
   - 放要打印的东西
   - [字符串]

只认英文

* ( ) , ; [ ]

*引号里面可中文，外须英文

P7 更多 print

1. 字符串连接 e.g. print("Hello" + " world " + "!")
   - 加空格
   - ⇒ Hello world!
2. 单双引号转义 e.g. print(' He said " good " ')
   - 配对

print("He said \"Let's go!\" ")

⇒ He said "Let's go!"

加右斜杠：后"只是单双引号

(右斜号)

3. 换行 e.g. print("Hello! \n Hi!")

\n：表示换行

4. 三引号跨行字符串 e.g. print(""" 君不见，…… 三引号

君不见""") ⇒ 君不见，…

君不见

""" """ 或 ''' ''' ：直换行

No.

Date

P8 python 变量（下面有取值顺序）要求：① 只能以字母/下划线_开头，且名称只能有字母、数字、_

eg. 先赋值：my_love = "1366…"

* 再用变量 print ("报打：" + my_love)

重新赋值 变量名 再赋值 my_ex = my_love

my_love = "1506…"

P9. 变量命名规则：只由文字、数字、下划线（_）组成 eg. zhang_3

① 不要用拼音 username ✓

② 下划线命名法：字母小写，用下划线分隔 user_age ~~user~~

③ 驼峰命名法：UserAge.

* 大小写敏感：user_age ≠ user_Age

* 不用关键字命名 eg. print、False

P10 python 数学运算

字符串	整数	浮点数
"6"	6	6.0

+ - * / (正斜杠)

顺序：( ) 乘方 */ +-

工具库：math (进数运算库，要先导入)

* import math

不能省 math.函数名(…) (功能) eg. math.sin()

不能省 print (math.log2(8))

或者 先赋值 result = math.log2(8)

print (result)

eg. b^2-4ac： b**(2) - 4*a*c

No.

Date

P11 Python 注释 # ... 简单行

整段 + #: command + / (再按取消)

或者 ''' 我. 不执行、无区别

怎

么'''

P12 Python 数据类型 ①字符串 str 方括号[索引]

只能用引 "Hello" "Hello"[3] 0123

len('Hello') 长度 → 5 提取该位置字符

*程序世界从0开始

0 1 2 3

② 整数 int ③ 浮点数 float

④ 布尔类型 bool 真 T很

大写 True False

⑤ 空值类型 NoneType

只一种值: None 0

""

False

→ type确定类型 eg. type("Hello") <class 'str'>

~ 命令行模式: 写好命令. 保存&运行 all type(6) <class 'int'>

P13 Python 交互式模型: 输一行 → 立即执行 → 看结果 终端

(命名不会 save. → 进行交互式环境: Python 控制台 - python:

一旦退出就 lose!!! *不需 print 功能就能直接看见结果

e.g. 计算器 eg. a. 3*2

→ 结束交互: quit()

No.

Date

P14 Python input：获取输入 input("提示")

一律返回字符串

eg. input("age: 25")

```
user_age = input("age:")
print("你今年" + user_age + "岁了!")
```

*转换类型：其他 → int ; 其他 → float ; 其他 → str

int("666") → 666

先转换再运算

eg.

```
user_age = int(input("age:"))
user_age_after_10 = user_age + 10
print("十年后" + str(user_age_after_10) + "岁了!")
```

P15 Python条件语句：如果 条件1 满足：执行 A行动 否则：执行 B行动

```
if [条件]:
    [执行语句]
else:
    [执行语句]
```

①为bool值：True / False ②比较运算：==、!=、>、<、>=、<=

缩进(4个空格)

P16 Python 嵌套 / 多条件判断：

```
if [条件1]:
    if [条件2]:
        [语句A]
    else:
        [语句B]
else:
    [语句C]
```

注意缩进

1为真 2为真 → 执行

1为真 2为假 → 执行

1、2为假 → 执行

多个条件：

```
if [条件1]:
    [语句A]
elif [条件B]:
    [语句B]
else:
    [语句C]
```

1为假后再判断

No.

Date

优先级:

P17 Python逻辑运算: ② and ③ or ① not

与 或 非

全True 一个True 取个 not x>5 True

↓ ↓ ↓

True True False

P18 Python列表: 数据结构: 整合关联的东西

空的表: shopping_list = [] 方括号

shopping_list = ["keyboard", "pen"]

应用方法: 加东西: shopping_list.append("显示器")

* 方法 函数 哪都能放

对象.方法名(...) 函数名(对象)

shopping_list.append("显示器") len(shopping_list)

* 可变 不可变 (变需再赋值)

列表 list、字典 dict str、bool、int、float ...

删东西: shopping_list.remove("显示器")

打印: print(shopping_list[0])

改东西: shopping_list[1] = "音响"

* print(max(list))

print(min(list))

print(sorted(list)) 排序的

P19 python字典: 人名与电话绑定起来 字典 dictionary

储存 键:值对

key value

查找

空字典: contacts = {} 花括号

No.

Date

对应

contacts = {"小明": "137000", 分隔
"小花": "136000"}

获取值：contacts["小明"] 不可变 key

元组：tuple ： ex_tuple = ("键盘", "键帽") 圆括，不可变. append. remove

list ： ex_list = ["键盘", "键帽"]

→ contacts = {("小明", 23): "137000",
用元组有3个→ ("小明", 37): "136000",
("小明", 39): "135000"}

更新
加值：contacts["小美"] = "186000"

检查："小明" in contacts → bool

key in dict

print ("小明" in contacts)

删除键值：del contacts["小明"]
key

多长！ len(contacts)

P20 Python for循环：迭代 * list. dictionary. str. 按顺序对里面的东西做一遍事

语句：for 变量名 in 可迭代对象：

缩进 → #对每个变量做一些事情

eg. for temp in temperature_list:
if temp >= 28:
print ("完了")

返回：temperature_dict.keys() #所有键

temperature_dict.values() #所有值

temperature_dict.items() #所有键值对

e.g. for staff_id, temp in temperature_dict:
赋值

*range 整数数列 ① range(5, 10) → 5. 6. 7. 8. 9

起始 结束

No.

Date

始 末 步长

② range (5, 10, 2)　　5, 7, 9

ex. 从1加到100:

```
total = 0
for i in range(1, 101):
    total = total + i
print(total)
```

*有循环次数

P21

Python while 循环　# measure_brightness 函数返回当前测量天空亮度

直到多久拍完?

```
if measure_brightness() >= 500:
    # 拍照片
    take_photo()
```

*条件何时结束未知　"一直做… 直到False"

结构: while 条件A:　→ bool: True 继续执行
　　行为B.

*total = total + num

简写: total += num　(自身加上num)

P22 Python 格式化字符串　e.g. 春节字幅

```
contacts = ["小美", "小帅", "小明", "小花"]
for name in contacts:
    message_content = name + "天天开心"
    send_message(name, message_content)
```

① format方法: message_content = """
金{0}岁, …
活{1}及家夫 …
""".format(year, name)

或 金{one}岁
活{two}~　代
""".format(one=…, two=…)

② f字符串: message_content = f"""
全{year} …
信{name} ..
"""

关键字: 变量

No.

Date

eg. gpa_dict = {"小明": 3.5, "小花": 3.8}

for name, gpa in gpa_dict.items():

print("{0}的gpa为{1}".format(name, gpa))

→ 保留n位小数: {1:.1f}".format(name, gpa))

:.1f (保留1位)

print(f"{name}的gpa为{gpa:.1f}")

P23 Python函数(上) DRY原则: Don't Repeat Yourself.

函数. eg. print. sum. type 制作机器 → 解决某一任务

*定义不会执行. 定义: def calculate_sector_1():

调用才会逐行执行) 关键字词: 开始定义 #定义代码

...

调用: calculate_sector_1()

通用: def calculate_sector(central_angle, radius): ↓输入. 自动执行

sector_area = central_angle / 360 * 3.14 * radius ** 2)

print(f"面积为{sector_area}")

return sector_area

eg. calculate_sector(30, 16), calculate_sector(15, 1)

P24. Python函数(下) *作用域: 函数def里: 局部变量、V访问

def外: 访问不到

return: def func():

a=3

print(a).

return a → 调用后返回a

No.

Date

常用

P25 Python 引入模块 . 内置函数 : python程序 * // 连除号：除完向下取整

eg. 模块: statistics 不常用

e.g. import statistics

print(statistics.median([69, 68, 70]))

方法: ① import 语句 : import statistics

② from … import … 语句

eg. from statistics import median, mean

模块 函数

print(median([60, 50]))

不推荐 ② from statistics import *

from … import * 全部引用

*第三方模块: 先安装+引入 pip install e.g. pypi.org

再引入 import

Java

P26 面向对象编程 Object Orrented Programming (oop): 模拟真实世界

↑ ~过程~ : 完成具体任务 C++ #定义ATM类 (模板)

class ATM:

→封装、继承、多态 def __init__(self, 编号)

(无序判断) self.编号 = 编号

创建2个ATM对象 (实例)

atm1 = ATM("001")

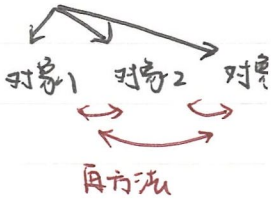
atm2 = ATM("002")

*对象: 属性 方法 任务 先分解

Attribute Method

拥有什么 能做什么

→ 放在类的变量 放在类的函数



No.

Date

P27 Python创建类(上)

定义属性

语构：class NameOfClass:

首字母大写

定义类：Pascal命名法：

e.g. UserAccount

#定义类代码

① class CuteCat:

def __init__(self):

#构造函数

self.name = "Lab"

cat1 = CuteCat()

print(cat1.name)

② class CuteCat:

def __init__(self, cat_name):

self.name = cat_name

cat1 = CuteCat("Jojo")

print(cat1.name)

P28 Python创建类(下)

定义方法

语构：class CuteCat:

def __init__(self, cat_name, cat_age):

self.name = cat_name

self.age = cat_age

① def speak(self):

print("喵" * self.age)

cat1 = CuteCat("Jojo", 2)

cat1.speak()

② def think(self, content):

print(f"猫{self.name}的见为{content}")

cat1.think("去玩")

No.

Date

P29 Python 类继承　　DRY：Don't Repeat Yourself.

{ 继承属性　people、cat ⇒ mammal　　父类
　～方法.　　语句：class Mammal:

　　def __init__(self, name):

* A是B　class A(B)　A是B子类　　self.name = name

eg. 人是动物　class 人(动物):　　def breathe(self):

　　print(self.name + "呼吸")

① class Human(Mammal):　　继承　子类　父类

　　def read(self):　继承+另外的

　　print(self.name + "阅读")

human1 = Human("人")　→父类的

human1.read()　→自己的

② 加:　def __init__(self, name):

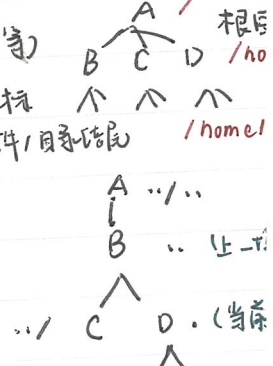
调用父类的属性，也有的　super().__init__(name)

　　self.has_tail = False

P30 Python 文件路径.　* 类 unix 操作系统 (linux, macos 等)

{ 绝对路径：从根目录出发：/ 开头，目标文件/目录所在

　相对路径：从一个参考位置出发



P31 Python 文件操作(1)

{ 打开文件：open("/usr/demo/data.txt", "r")

　"r" 读取模式 (只读)，→ 默认

　"w"、写入模式 (只写)

　e.g. open("/data.txt", "r", encoding = "utf

　　编码方式

No.

Date

* read方法: f = open("/data.txt", "r", encoding

返回str ← ① print(f.read()) #读完打印="utf-8")

print(f.read()) #返回空,读完了

② print(f.read(10)) #读1-10字节

③ print(f.readline()) #读1行 根据/n

④ line = f.readline()

while line != "":

print(line)

line = f.readline() #下一行

⑤ print(f.readlines()) 读取全部

*关闭文件: ① f.close() ← 关闭,释放

② with open("./data.txt") as f: 自动关闭

print(f.read())

P32 Python文件操作(下) 写文件: with open("./unknown.txt", "w") as f: w写

清空再写

f.write("Hello!")

f.write("Yoooo")

> Hello!Yoooo 不默认换行,\n

原文件加内容: with open("./data.txt", "a") as f:

a附加模式

读+写文件: with open("./data.txt", "r+") as f:

r+支持读+写

P33 Python异常处理 异常类型 { IndexError 索引错误

ZeroDivisionError 除零错误

FileNotFoundError 找不到文件错误

TypeError 类型错误

↓测试 ...

debug

No.

Date

*捕捉异常：try - except 语句

```
try:
    #可能报错的报码
① except ValueError:                    ↓ 一步
    print("输入不合理。").
  except ZeroDivisionError:             ↓ 一步
    print("...")
② exp except:          → 所有错误       ↓ 运行
    print("未知错误")
  else:                 → 无错误
    print("正确")
  finally:              → 无论有/无错都输出
    print("结束")
```

P34 Python 测试(上)：assert 语句 + bool (True/False)

eg. assert len("Hi") == 2

① True：无事发生，继续运行

② False：AssertionError，中止程序.

*unittest：单元测试库：验证 min 单元

eg. import unittest

划分 { 实现代码

测试代码

测试思路

引入 from my-calculator import my-add

类 class TestMyAdder(unittest.TestCase.

开头比 def test_p_with_p(self):

self.assertEqual(my-adder(5,3),8

No.

Date

P35 Python 测试(下) * unittest.TestCase 类的常见测试方法

assertEqual(A,B) → 判断 assert A == B

assertTrue(A) False assert A is True

assertIn(A,B) NotIn assert A in B

结构: import unittest

from sentence import Sentence

class TestSentence(unittest.TestCase):

def setUp(self):

都运行一次 避免重复 小 大 self.sentence = Sentence("Hello")

* 测试. terminal 输入: python -m unittest 文件名 shopping_list

P36 Python 高阶函数和匿名函数 * 计算函数当作参数: 高阶函数

eg. calculate_and_print(3, calculate_square) 传入函数本身,不调用

* 只用一次的函数: 匿名函数 ~~lamda~~ lambda 直接写结果.

eg. calculate_and_print(7, lambda num: num 不return,

print_with_vertical_bar)

匿名 参数

*增加参数: lambda num1 num2:

num1 + num2

→调用: (lambda num1, num2: num1 + num2)(2, 3)

(最简单) 加参数.