Python 快速入门

一节简洁的 Python 入门课程

开发部下学期授课规划

主讲**后端内容**，使网页具备实际功能，能够处理数据。

- Python 语言
- Django 框架（Python常用后端框架）
- 数据库和SQL——存储数据的地方
- 综合设计

上学期回顾

- 学习了前端知识，包含了HTML CSS JS，了解如何设计出与用户进行交互的用户界面。
- 完成了大作业，综合一学期学到的知识，也有一些收获。

选修课推荐：**《Web前端开发基础》**

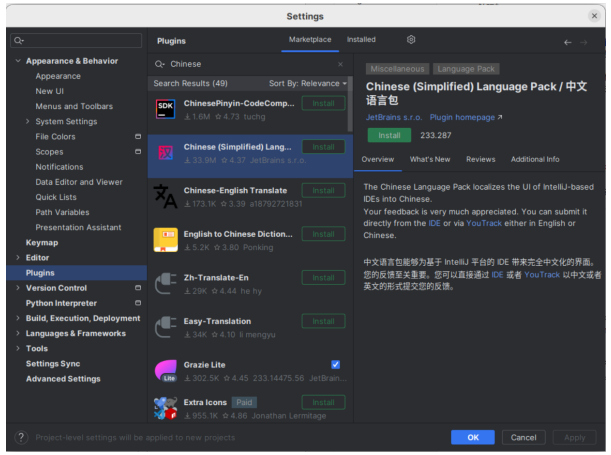
本节课内容

1. Python 语法概览
2. 变量定义和使用
3. 循环和控制
4. 常用数据结构
5. 函数
6. 类和面向对象
7. 常用库和函数介绍

学完本节课，你将可以……

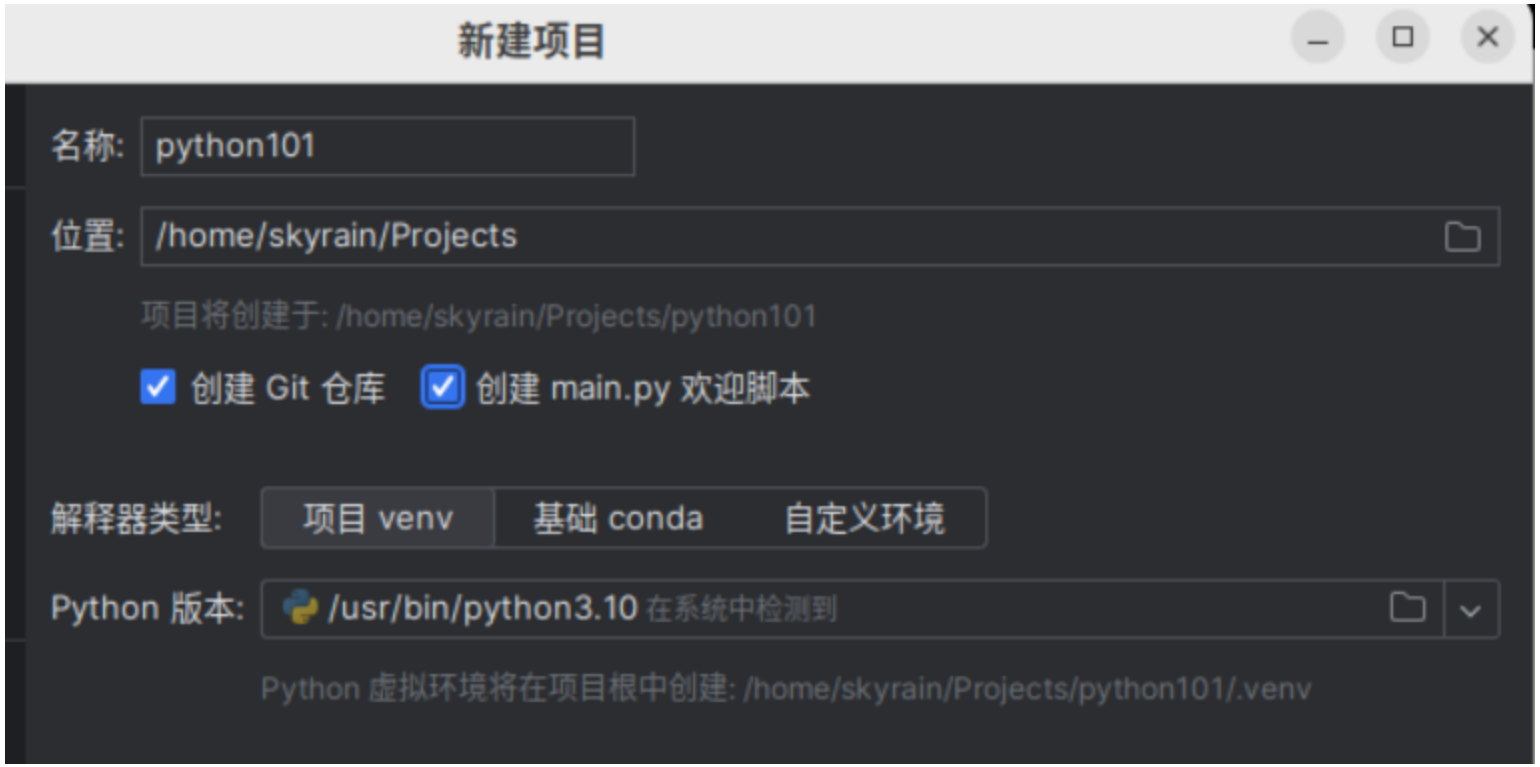
- 报考计算机二级 Python *蓝桥杯有点勉强*
- 使用 Python 语言**制作爬虫** *爬取新闻 图片 视频等*
- 利用 Python 语言**解决生活小问题** *批量重命名 查找文件内容 给图片加水印*
- **学校的课程** 计算机基础 选修课《小白玩转数据》等和 Python 有关的课程将变得更加轻松

Python 环境安装

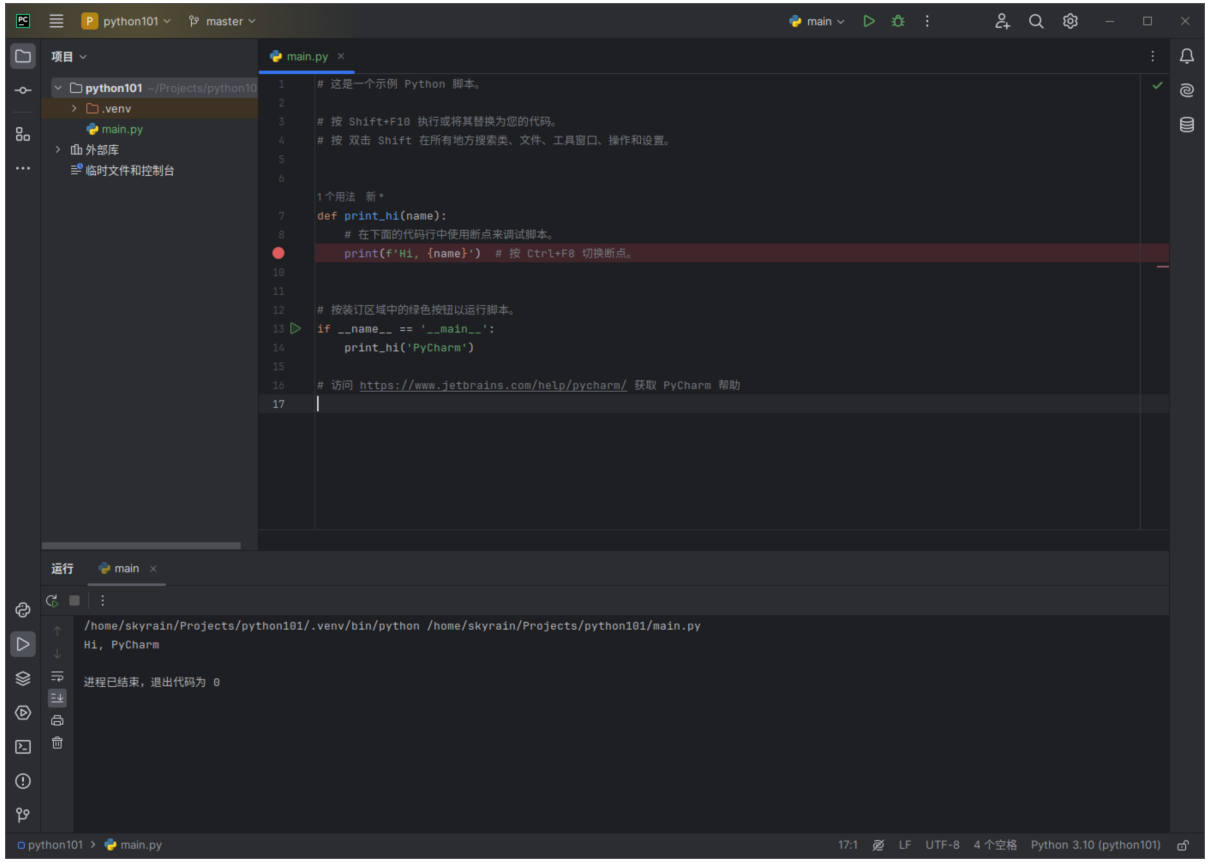
- 打开 PyCharm Professional 下载页面，下载IDE并安装。
- 进入欢迎页面后，点击 Customize - All settings, 打开设置界面
- 点击 Plugins，搜索 Chinese 安装简体中文语言包，安装完成后点击 Restart 重启 IDE
- 

新建 Python 项目

- 在欢迎页面点击新建 Python 项目，选择“纯Python”
- 名称随便写，勾选 “创建 Git 仓库”和“创建欢迎脚本”
- 解释器类型选择“项目venv”



- 一样后点击“确定”创建新项目，打开之后稍等片刻



- 左侧显示项目内容，右侧显示当前打开的代码，下方是终端
- 每做一个新练习，在左侧的文件夹右键——新建——Python文件，在新建的Python文件中编辑，按`Ctrl+Shift+F10`运行当前文件(或右键菜单——运行当前文件)

Python 介绍

```
print('Hello world')
```

这是一段Python代码。将它保存在名为`hello.py`的文件中，输入`python3 hello.py`命令即可运行。

Python 是一种**解释性语言**，不需要编译，直接输入到解释器即可执行。和JS差不多。

基本语法元素

- 一个 `py` 文件就是一个 Python 程序
- 使用**四个空格或TAB的缩进**来指示每行语句之间的关系（JS如何指示？）
- 行尾无须分隔符，如需要在一行内写下多行程序，可用 `;` 分隔

```
import random
rand = random.randint(1, 100)
guess = 0
while True:
    guess = int(input('请你猜一个1-100之间的数: '))
    if guess == rand:
        print('你猜对啦! ')
        break
    else:
        print('猜错了，再试一次吧')
    pass
```

关于 Python 程序格式框架的描述，以下选项中错误的是

1. Python 语言的缩进可以采用 Tab 键实现
2. Python 单层缩进代码属于之前最邻近的一行非缩进代码，多层缩进代码根据缩进关系决定所属范围
3. 判断、循环、函数等语法形式能够通过缩进包含一批 Python 代码，进而表达对应的语义
4. Python 语言不采用严格的“缩进”来表明程序的格式框架

注释

- 单行注释以 `#` 开头，`#`之后的此行内容都会被忽略
- 多行注释前后被`'''`包裹
- 函数签名和类签名后紧跟的多行注释被视为文档注释

```
import random # 导入库 random
rand = random.randint(1, 100) # 生成一个1-100指尖的随机数
guess = 0
while True: # 进行循环
    guess = int(input('请你猜一个1-100之间的数: '))
    if guess == rand: # 判断是否猜对
        print('你猜对啦! ') # win
        break
    else:
        print('猜错了，再试一次吧') # lose
    pass
```

关于 Python 语言的注释，以下选项中描述错误的是

1. Python 语言的单行注释以 `#` 开头
2. Python 语言的单行注释以单引号 `'` 开头
3. Python 语言的多行注释以 `'''` （三个单引号）开头和结尾
4. Python 语言有两种注释方式：单行注释和多行注释

变量

- 使用等号声明变量和给变量赋值
- 变量的命名一般采用下划线命名法，如`yanshan_university` `text_response` `i_love_python`等
- 没有常量，但习惯用全大写下划线命名来表示常量，如`SYSTEM_MAX_NUMBER` `USERNAME`

```
TARGET_NUMBER = 503 # 常量
guess_number = int(input('猜一个数: ')) # 定义变量
result = False # 定义变量
if TARGET_NUMBER == guess_number:
    print('你猜对了')
    result = True # 变量赋值
else:
    print('你猜错了TT')
    result = False
```

变量的命名

规则：

1. 第一个字符必须是字母或下划线
2. 命名的其他部分可以由字母、下划线和数字组成
3. 命名对大小写敏感
4. 可以使用中文命名，但不推荐。当使用中文命名时第一个字符可以是中文、字母和下划线
5. 保留字不得用于变量的命名
6. 当你不想使用一个变量时，给它取名为 `_`

```
hello_fr0m_yuna = 'World'
# ?hi = 1  此行出错
_hello = 'World'
中文变量 = '不推荐'
```

以下选项中符合Python语言变量命名规则的是

1. `*i`
2. `3_1`
3. `Al!`
4. `Templist`

保留字

- 保留字是Python内部使用的关键字，不能用来作为程序中的名字

```
>>> import keyword
>>> keyword.kwlist
['False', 'None', 'True', 'and', 'as',
'assert', 'break', 'class', 'continue', 'def',
'del', 'elif', 'else', 'except', 'finally', 'for',
 'from', 'global', 'if', 'import', 'in', 'is',
 'lambda', 'nonlocal', 'not', 'or', 'pass',
 'raise', 'return', 'try', 'while', 'with', 'yield']
```

上面的代码列出了Python中的保留字

以下选项中不是 Python 语言的保留字的是

1. `except`
2. `do`
3. `pass`
4. `while`

变量的赋值

- 使用`=`进行赋值，若左侧的变量名未找到，则视为定义了一个新变量，变量的值是等号右侧值
- 可以进行连续赋值，如`a = b = c = 114514`
- `,` 可以用于高级赋值，如`a, b = b, a`交换两个变量的值，`name, addr = ('Wang', 'YSU')` 用于拆分赋值元组中的元素
- 变量的类型是动态的，可以变化变量的类型。可以使用`type()`函数来确定变量当前的类型。

```
name, phone = 'SkyRain', 10086
type(phone) == int # True
phone = '10086'
type(phone) == str # True
```

关于赋值语句，以下选项中描述错误的是

1. 在 Python 语言中，有一种赋值语句，可以同时给多个变量赋值
2. 设 `x = "alice"; y = "kate"`，执行 `x,y = y,x` 可以实现变量 x 和 y 值的互换
3. 设 `a = 10; b = 20`，执行 `a,b = a,a + b; print(a,b)` 和 `a = b; b = a + b;print(a,b)` 之后，得到同样的输出结果：`10 30`
4. 在 Python 语言中，`=` 表示赋值，即将 `=` 右侧的计算结果赋值给左侧变量，包含 `=` 的语句称为赋值语句

基本数据类型

- 数字：`int` 代表整数 `float` 代表浮点数
- 布尔值：`bool` 布尔值，取值为`True`和`False`
- 字符串：可用一对单引号或双引号定义一个单行字符串，用一对`'''`定义多行字符串。
- 列表：`list` 存储一些值
- 元组：`tuple` 和列表类似，但创建后不可变
- 字典：`dict` 键值对存储
- 集合：`set` 内部元素保持唯一性
- `None` 用于表示空值

整数

```
num_1 = 123 # 这是一个整数
num_2 = 456 # 这也是一个整数
num_3 = 10e100 # 好大
num_4 = 0x114514 + 0b0100 + 0o1234 # 支持十六进制和八进制和二进制
num_5 = int('114514') # int() 函数可以将其他类型转换为int类型
```

- Python中整数可以保存无限位数的数，所以Python算法比赛不会考高精度算术算法

浮点数和复数

```
fl_1 = 0.0
fl_2 = 0.1
fl_3 = 0.2
0.1+0.2 == 0.3 # False
fl_4 = float('114.514') # 使用 float() 函数将其他类型转换为float类型

comp = -.114514+6j # 原生支持复数
```

- Python的float**存在误差**，因此python中`0.1+0.2`不等于`0.3`
- 复数一般用不到

数字运算符和逻辑运算符

- `+ - * \` 用于算术
- `\\` 用于整数除法
- `%` 用于取余
- `**` 表示幂，如`2**2==4`
- `& ^ | << >>`用于位运算

逻辑运算符

- `and` 逻辑与 `or` 逻辑或 `==` 相等 `> >= <= <` 等等
- `is` 用于判断是否为空，如`l is None`

`math`库中存有很多和数学运算相关的函数，如`abs()` `ceil()` `floor()`

关于 Python 的数字类型，以下选项中描述错误的是

1. Python 整数类型提供了 4 种进制表示：十进制、二进制、八进制和十六进制
2. Python 语言要求所有浮点数必须带有小数部分
3. Python 语言中，复数类型中实数部分和虚数部分的数值都是浮点类型，复数的虚数部分通过后缀“C”或者“c”来表示
4. Python 语言提供 int、float、complex 等数字类型

以下选项中，属于Python语言中合法的二进制整数是

1. 0B1010
2. 0B1019
3. 0bC3F
4. 0b1708

关于Python语言数值操作符，以下选项中描述错误的是

1. x//y表示x与y之整数商，即不大于x与y之商的最大整数
2. x**y表示x的y次幂，其中，y必须是整数
3. x%y表示x与y之商的余数，也称为模运算
4. x/y表示x与y之商

关于Python语言的浮点数类型，以下选项中描述错误的是

1. 浮点数类型表示带有小数的类型
2. Python语言要求所有浮点数必须带有小数部分
3. 小数部分不可以为0
4. 浮点数类型与数学中实数的概念一致

str 字符串

字符串有很多属性。可以使用`dir()`函数列出一个对象或类的属性：

['capitalize', 'casefold', 'center', 'count', 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha', 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace', 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix', 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split', 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']
此外，str还支持下标运算符，接下来我们会讲解。

列表和元组

- `[]` 声明一个列表
- `len()` 函数可以查询列表的长度
- `append()` 函数可以向列表末尾添加元素
- `insert()` 函数可以向列表的任意位置添加元素
- `pop()` 从列表开头弹出一个元素
- `[]` 可以获取一个或多个元素

```
l = [1, 2, 3, 4, 5]
len(l) # 5
l.append(6)
r = l.pop() # r=1
print(l[0]) # 2
```

使用 `()` 定义元组，元组和列表的区别是元组在创建后不能被改变（添加、删除元素）

列表

- `in` 关键字用于查询列表中是否存在特定元素

```
>>> l = [1, 2, 3, 4]
>>> 1 in l
True
>>> 5 in l
False
```

- `del` 关键字用于删除列表中的一个元素

```
>>> l = [1, 2, 3, 4]
>>> del l[0]
>>> l
[2, 3, 4]
```

[] 的用法

基础用法：`l[i]` 能够从l中选取第i个元素
完全格式：`l[start:stop:step]`

- start指从第几个元素开始选
- stop指选到第几个元素（不包含此元素）
- step指每选一个元素就跳过几个元素
- 完全格式将返回一个l的子列表

例子:

```
>>> l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> l[0]
1
>>> l[:10]
[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
>>> l[-1]
10
>>> l[::-1]
[10, 9, 8, 7, 6, 5, 4, 3, 2, 1]
>>> l[2:5]
[3, 4, 5]
>>> l[1:8:2]
[2, 4, 6, 8]
```

下面代码的执行结果是

1

```
ls=[[1,2,3],[[4,5],6],[7,8]]
print(len(ls[0]))
```

2

```
ls = ["2020", "20.20", "Python"]
ls.append(2020)
ls.append([2020, "2020"])
ls += [2024, ]
print(ls)
```

关于 Python 序列类型的通用操作符和函数，以下选项中描述错误的是

1. 如果 `x` 不是 `s` 的元素，`x not in s` 返回 `True`
2. 如果 `s` 是一个序列，`s = [1,"kate", True]，s[3]` 返回 `True`
3. 如果 `s` 是一个序列，`s = [1,"kate",True]，s[-1]` 返回 `True`
4. 如果 `x` 是 `s` 的元素，`x in s` 返回 `True`

给出如下代码：

```
TempStr = "Hello World"
```

以下选项中可以输出"World"子串的是

1. `print(TempStr[-5: -1])`
2. `print(TempStr[-5:0])`
3. `print(TempStr[-4: -1])`
4. `print(TempStr[-5:])`

dict 字典

字典是一种键值对数据结构。

- `{}` 定义一个字典，键值对用`:`分隔
- `[]` 可用于在字典中获取值
- `keys()` 函数用于返回一个字典中所有键的迭代器
- `values()` 函数用于返回一个字典中所有值的迭代器
- `items()` 函数返回一个字典中键值对的迭代器
- `get(key, default)` 函数在找不到key时返回default，而不是像下标运算符那样抛出错误

给出如下代码

```
MonthandFlower={“1月”:“梅花”,“2月”:“杏花”,“3月”:“桃花”,“4月”:“牡丹花”,
“5月”:“石榴花”,“6月”:“莲花”,“7月”:“玉簪花”,“8月”:“桂花”,“9月”:“菊花”,
“10月”:“芙蓉花”,“11月”:“山茶花”,“12月”:“水仙花”}

n = input(“请输入1—12的月份:”)
print(n + “月份之代表花: ” + MonthandFlower.get(str(n)+“月”))
```

1. 代码实现了获取一个整数（1—12）来表示月份，输出该月份对应的代表花名
2. MonthandFlower是列表类型变量
3. MonthandFlower是一个元组
4. MonthandFlower是集合类型变量

控制结构

- `if` 结构
- `match` 结构（Python 3.10）新增

if 语句

```
if condition:
    # True
elif condition:
    # True
elif condition:
    # True
else:
    # True
```

`condition` 为一个布尔值表达式。 `elif` 和 `else` 为可选

match 语句

Python 3.10 新增。行为类似其他语言中的 `switch`

```
match [1, 2, 3, 4].pop():
    case 1:
        print('1')
    case 2:
        print('2')
    case _: # default
        print('Default')
        pass
    pass
```

循环语句

- `for` 语句用来遍历迭代器
- `while` 语句用来在指定条件下循环

什么是迭代器?

- 一个内部包含元素的对象
- 能够获取当前的元素值
- 能够切换到下一个元素

Python中很多函数都接受迭代器，`for`语句也不例外。从迭代器的定义知道，`list`是一个迭代器。

for 循环

```
for var in iterable:
    # do something
    pass
```

for 循环用来遍历迭代器，每次循环中var都会被赋值为当前的元素

```
>>> for i in [1, 2, 3]:
...     print(i)
...     pass
1
2
3
```

while 循环

```
while condition:
    # do something
    pass
```

while 循环一直执行，直到`condition`为`False`

```
>>> i = 3
>>> while i > 0:
...     print(i)
...     i -= 1
...     pass
3
2
1
```

循环内控制

- `break` 用于终止循环
- `continue` 用于跳过这次循环，直接进行下次循环

这两个关键字只能对最内层循环起作用

下面代码的输出结果是

```
for s in "HelloWorld":
    if s=="W":
        break
    print(s, end="")
```

关于分支结构，以下选项中描述不正确的是

1. if 语句中条件部分可以使用任何能够产生True和False的语句和函数
2. 二分支结构有一种紧凑形式，使用保留字if和elif实现
3. 多分支结构用于设置多个判断条件以及对应的多条执行路径
4. if 语句中语句块执行与否依赖于条件判断

什么输入能够结束程序运行?

```
while True:
    guess = eval(input())
    if guess == 0x452//2:
        break
```

`eval(str)` 函数将输入的字符串当作Python代码执行并返回结果。

下面代码的输出结果

```
for s in "abc":
    for i in range(3):
        print (s,end="")
        if s=="c":
            break
```

`range(start, end)` 是一个内置函数，用来生成从 `start` 到 `end` 的整数序列迭代器。

下面代码的输出结果

```
for i in range(10):
    if i%2==0:
        continue
    else:
        print(i, end=",")
```

编写程序，键盘输入一段文本，保存在一个字符串变量s中，倒序输出此字符串并统计字符串中字符数量

从键盘输入四个数字，各个数字采用空格分隔，对应变量为x0, y0, x1, y1。计算两点(x0, y0) (x1, y1)之间的距离，输出这个距离。

键盘输入一个字符串s，要求按下面的格式将s输出到屏幕。宽第为20个字符，等号字符填充，居中对齐。如果输入字符串超过20位，则输出前20位。

手疼，不写了，剩下的自由发挥

函数 全局变量和局部变量 面向对象

函数和类

```
def isEven(number):
    if number >= 0 and number % 2 == 0:
        return True
    else:
        return False
    pass

isOdd(2)
```

- `def` 声明一个函数，格式为 `def 函数名(参数1, 参数2, 参数3...):`
- `return someValue` 在函数中返回一个值

全局变量和局部变量

```
name = 'skyrain'

def say_hello():
    # global name
    print(f'Hello, {name}')
    pass

say_hello()
```

- 函数体中默认访问不到全局变量
- 使用 `global` 关键字+变量名以声明全局变量
- 局部变量在函数执行完后释放，全局变量不会

匿名函数

```
func = lambda x: print(f'Hello, {x}')

func('skyrain')
```

- `lambda` 定义一个匿名函数，格式是`lambda 参数1, 参数2...: 语句`
- 常用于 排序函数`sorted()` 映射函数`map()`

函数是可以嵌套的

```
def hello(name):
    def hi():
        print(f'Hi, {name}')
        pass
    return hi

hello('skyrain')()
```

上面的代码会正常运行

类

```
class YUNA:

    def __init__(self, arg1, arg2...):
        # do something
        pass

    pass
```

- `class` 关键字定义一个类
- 类包含属性和函数（成员函数、静态函数）
- 成员函数必须有第一个参数`self`，指向对象
- 在函数前加入`@staticmethod`构成一个静态函数，静态函数可无参数
- 函数名带有双下划线即为Python魔法函数，具有特殊作用，如`__init__`是构造函数

类和对象

```
class YUNA:

    def __init__(self, names):
        self.names = names
        print(f'Hi members: {names}')
        pass

    def say_hi_to(self, name):
        if name in self.names:
            print(f'Hi {name}')
    pass

y1 = YUNA(['skyrain', 'lyf', 'Trisuyan', 'yjh'])
y1.say_hi_to('lyf')
```