

BUSS 3620.人工智能导论

Python回顾

刘佳璐

安泰经济与管理学院

上海交通大学

目标

- 更好的掌握课上所学知识
 - 完成大作业
- 学习/回顾Python
 - Python有许多AI相关的包(library)
 - 易读
 - 易写
- **内容不限定某种编程语言**
 - 未来一种新的编程语言可能会更受欢迎
 - 学习如何自学新语言

为什么要学习编程?

- 最简单的让世界变得更好的方法，特别是对于没有资源的人
 - 一个人就可以
 - 一台电脑就行
 - 有实际影响



BUSS 3620.人工智能导论

编程语言均包含的一些要素

刘佳璐

安泰经济与管理学院

上海交通大学

编程语言均包含的一些要素

- 图灵机

- 示例：在第一个“0”之前是否有偶数个“1”？

- 1：是

- 0：否

当前状态Current State:		
状态State	输入	动作
偶数 开始	1	赋值 State=“奇数”, 向右移动
偶数	0	写 1,赋值 State = “中止”
奇数	1	赋值 State=“偶数”,向右移动
奇数	0	写0,赋值 State = “中止”

读

写



内存磁带

编程语言均包含的一些要素

- 伪代码(Pseudocode)
 - 问题:在第一个 “0”之前是否有偶数个 “1” ?
 - 输出: 1: 是; 0: 否

当前状态Current State:		
状态State	输入	动作
偶数/开始	1	赋值 State=“奇数”, 向右移动
偶数	0	写 1, 赋值 State = “中止”
奇数	1	赋值State=“偶数”,向右移动
奇数	0	写0,赋值State = “中止”

```
1  赋值current state = “偶数”
2  读取内存磁带的第一个数字
3  如果 current state 是 “偶数”
4      如果读取的数是 1
5          赋值current state = “奇数”
6          读取内存磁带中的下一个数字
7          返回 Line 3
8      否则
9          写 1
10         程序终止
11 否则
12     如果读取的数是1
13         赋值current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18         程序终止
```

函数 Functions

- 动词或动作
 - 解决较小的问题

```
1 赋值 current state = “偶数”
2 读取内存磁带的第一个数字
3 如果 current state 是 “偶数”
4     如果读取的数是 1
5         赋值 current state = “奇数”
6         读取内存磁带中的下一个数字
7         返回 Line 3
8     否则
9         写 1
10        程序终止
11 否则
12     如果读取的数是1
13         赋值 current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18        程序终止
```

变量 Variables

- 存储

- 文本
- 数字

```
1 赋值 current state = “偶数”
2 读取内存磁带的第一个数字
3 如果 current state 是 “偶数”
4     如果读取的数是 1
5         赋值 current state = “奇数”
6         读取内存磁带中的下一个数字
7         返回 Line 3
8     否则
9         写 1
10        程序终止
11 否则
12     如果读取的数是1
13         赋值 current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18        程序终止
```


条件 Conditionals

- 如果.....就.....否则.....
 - 不同的路径

```
1 赋值 current state = “偶数”
2 读取内存磁带的第一个数字
3 如果 current state 是 “偶数”
4     如果读取的数是 1
5         赋值current state = “奇数”
6         读取内存磁带中的下一个数字
7         返回 Line 3
8     否则
9         写 1
10        程序终止
11 否则
12     如果读取的数是1
13         赋值current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18        程序终止
```

判断 Boolean expression

- True或False

- 是或否
- 1或0
- 要满足的条件

```
1 赋值 current state = “偶数”
2 读取内存磁带的第一个数字
3 如果 current state 是 “偶数”
4     如果读取的数是 1
5         赋值current state = “奇数”
6         读取内存磁带中的下一个数字
7         返回 Line 3
8     否则
9         写 1
10        程序终止
11 否则
12     如果读取的数是1
13         赋值current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18        程序终止
```

循环 Loop

- 重复做一件事

```
1 赋值 current state = “偶数”
2 读取内存磁带的第一个数字
3 如果 current state 是 “偶数”
4     如果读取的数是 1
5         赋值current state = “奇数”
6         读取内存磁带中的下一个数字
7         返回 Line 3
8     否则
9         写 1
10        程序终止
11 否则
12     如果读取的数是1
13         赋值current state = “偶数”
14         读取内存磁带中的下一个数字
15         返回 Line 3
16     否则
17         写 0
18        程序终止
```

编程语言均包含的一些要素

- 函数 Functions
- 变量 Variables
- 条件 Conditionals
- 判断 Boolean expressions
- 循环 Loops

有问题吗？

- 请随时举手提问。



BUSS 3620.人工智能导论

Python前期准备

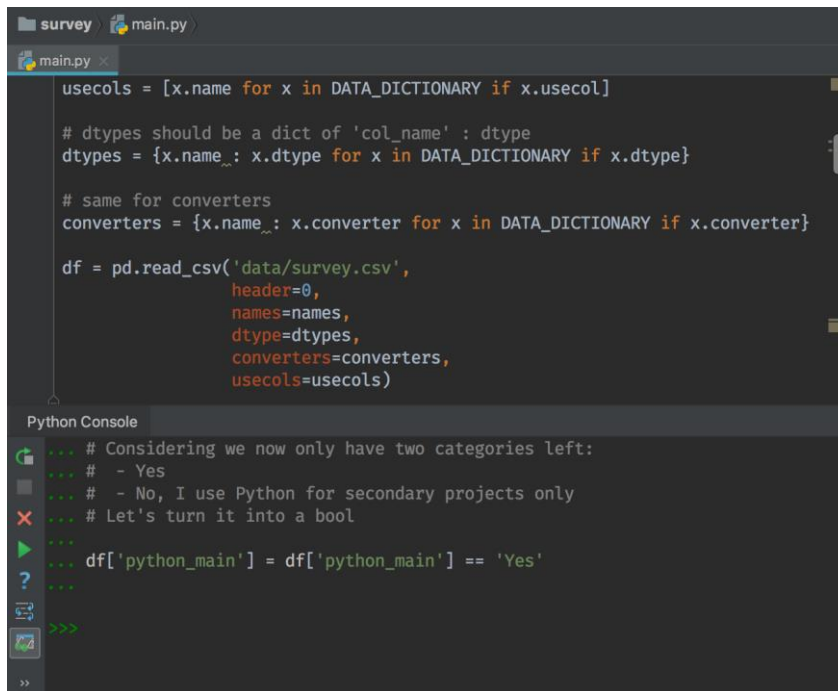
刘佳璐

安泰经济与管理学院

上海交通大学

集成开发环境 (IDE)

- Pycharm
 - 一整个程序 (AI)



```
survey > main.py
main.py
usecols = [x.name for x in DATA_DICTIONARY if x.usecol]

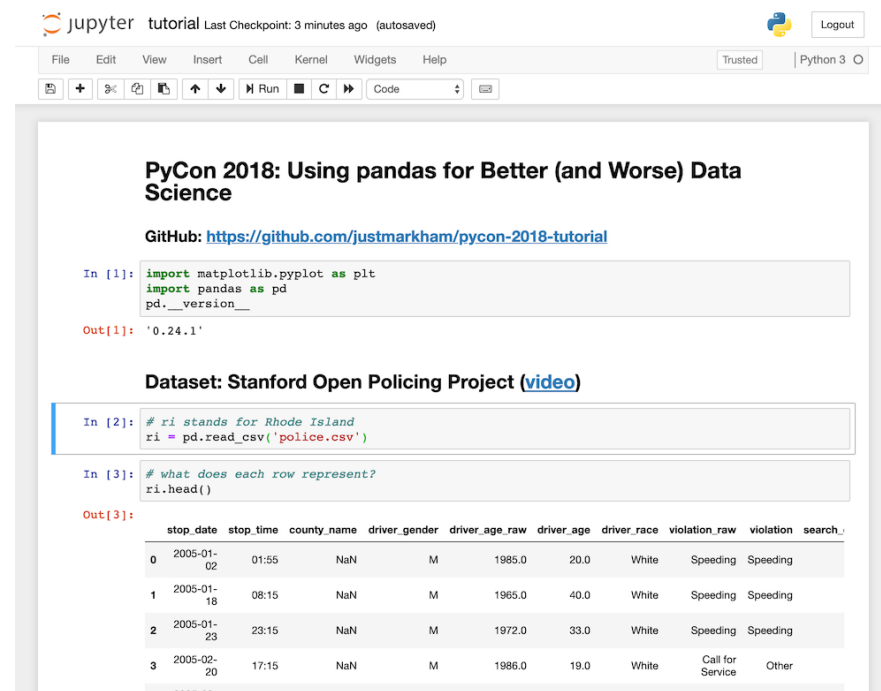
# dtypes should be a dict of 'col_name' : dtype
dtypes = {x.name_: x.dtype for x in DATA_DICTIONARY if x.dtype}

# same for converters
converters = {x.name_: x.converter for x in DATA_DICTIONARY if x.converter}

df = pd.read_csv('data/survey.csv',
                 header=0,
                 names=names,
                 dtype=dtypes,
                 converters=converters,
                 usecols=usecols)

Python Console
... # Considering we now only have two categories left:
... # - Yes
... # - No, I use Python for secondary projects only
... # Let's turn it into a bool
...
... df['python_main'] = df['python_main'] == 'Yes'
...
>>>
```

- Jupyter notebook
 - 一行一行运行 (数据分析)



jupyter tutorial Last Checkpoint: 3 minutes ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

PyCon 2018: Using pandas for Better (and Worse) Data Science

GitHub: <https://github.com/justmarkham/pycon-2018-tutorial>

```
In [1]: import matplotlib.pyplot as plt
import pandas as pd
pd.__version__

Out[1]: '0.24.1'
```

Dataset: Stanford Open Policing Project ([video](#))

```
In [2]: # ri stands for Rhode Island
ri = pd.read_csv('police.csv')

In [3]: # what does each row represent?
ri.head()

Out[3]:
```

	stop_date	stop_time	county_name	driver_gender	driver_age_raw	driver_age	driver_race	violation_raw	violation	search...
0	2005-01-02	01:55	NaN	M	1985.0	20.0	White	Speeding	Speeding	
1	2005-01-18	08:15	NaN	M	1965.0	40.0	White	Speeding	Speeding	
2	2005-01-23	23:15	NaN	M	1972.0	33.0	White	Speeding	Speeding	
3	2005-02-20	17:15	NaN	M	1986.0	19.0	White	Call for Service	Other	

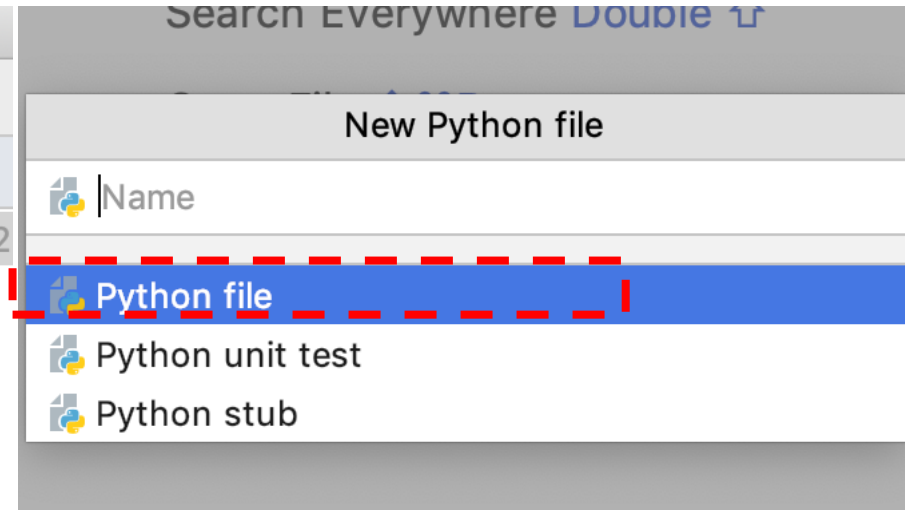
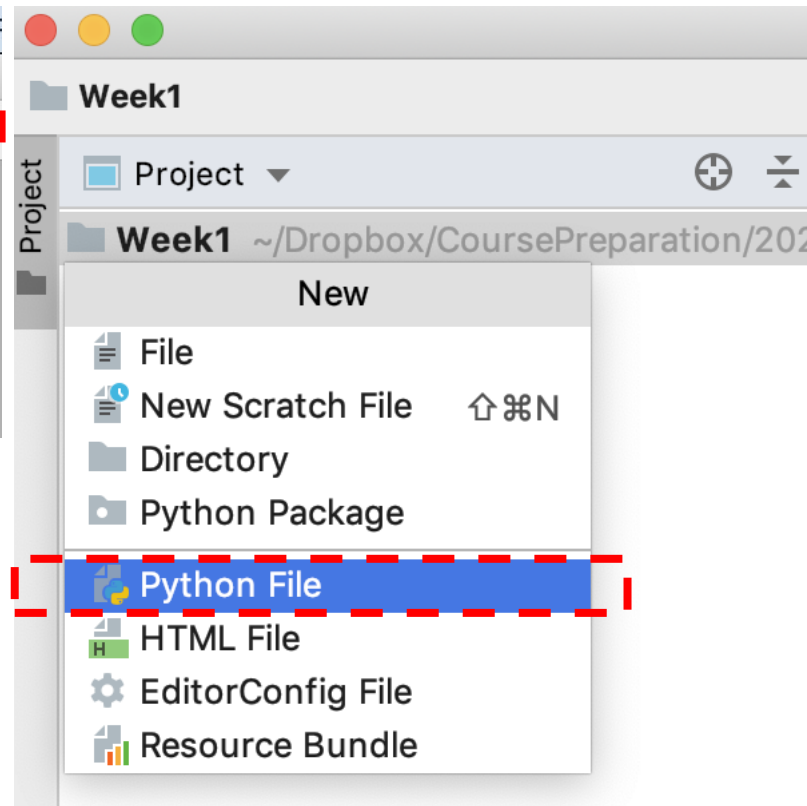
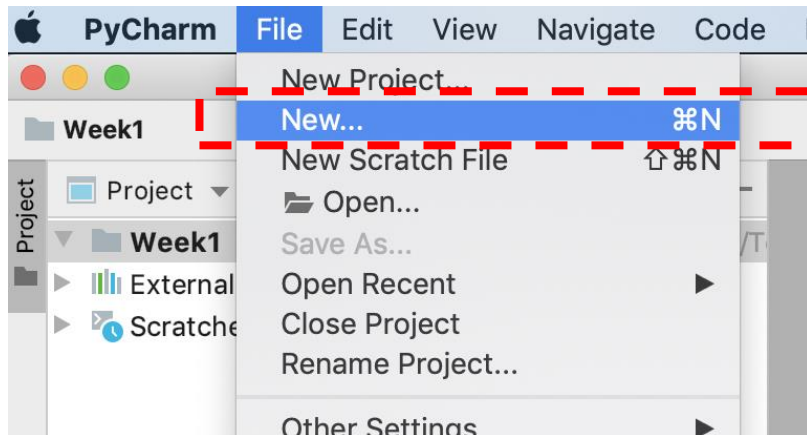
集成开发环境 (IDE)

- PyCharm
- 编写、运行代码



创建一个 python 文件 (.py)

- File → New → Python File



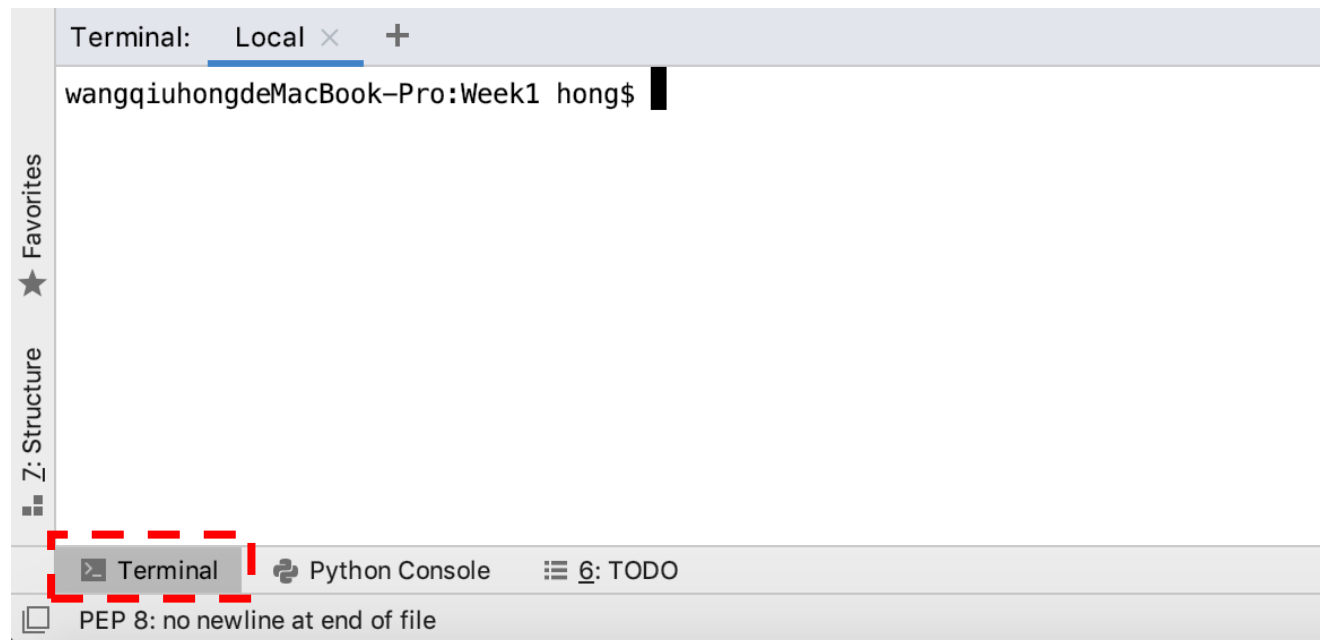
互动模式 Interactive mode

- 运行py文件中的**一行**代码
 - Mac: command + Enter (⌘ + ↵)
 - Windows: ALT+SHIFT+E
- 更改键盘快捷键：
 - Mac: PyCharm -> Preference -> Keymap – Other -> Execute selection in Python Console
 - Windows: File -> Settings -> Keymap – Other -> Execute selection in Python Console

脚本模式 Script mode

- 运行py文件中的**所有**代码
 - 打开 终端/命令行(Terminal)

```
python hello.py
```



命令行交互(Command-line interface)

- 列出当前文件夹中的文件

```
ls
```

- 创建一个新文件夹

```
mkdir directoryName
```

- 转到文件夹

```
cd directoryName
```

- 创建一个新的 python 文件

```
Mac: touch pythonFileName.py  
Windows: notepad pythonFileName.py
```

- 返回上一级文件夹

```
cd ..
```

练习 #1

- 从课程中心平台Canvas上下载作业0单元中的Homework0.zip并且解压缩
- 转到这个文件夹
- 列出这个文件夹里所有文件
- 运行里面的python文件

有问题吗？

- 请随时举手提问。



BUSS 3620.人工智能导论

Python

刘佳璐

安泰经济与管理学院

上海交通大学

报错 Exceptions

- 代码中的错误
 - NotImplementedError
 - ...
 - <https://docs.python.org/3/library/exceptions.html>
 - 可以自己发起报错

`raise Error`

```
BaseException
+-- KeyboardInterrupt
+-- Exception
    +-- ArithmeticError
    |   +-- ZeroDivisionError
+-- AssertionError
+-- AttributeError
+-- EOFError
+-- ImportError
    +-- ModuleNotFoundError
+-- LookupError
    +-- KeyError
+-- NameError
+-- SyntaxError
    +-- IndentationError
+-- ValueError
...

```


练习 #2

- 用print函数时，如果没有括号会怎么样？如果print和括号之间有空格会怎么样？
- 数学中，数字可以0开头，python可以吗？

报错 Exceptions

试一下代码有没有错

```
try:  
    Some code  
except:  
    Some code
```

```
1  try:  
2      1/0  
3  except:  
4      print('Wrong')  
5  
6  
7  try:  
8      1/0  
9  except ZeroDivisionError:  
10     print('Wrong')
```

注释 Comments

- 添加注释

#

- 说明代码的目的
- 不运行这一段代码

```
12  ## This part cannot run properly because 1/0 is not a value error.|
13  try:
14      1/0
15  except ValueError:
16      print('Wrong')
```

- 注释已有的代码块

- Mac: command + slash (⌘ + /)
- Windows: ctrl + shift + /

练习 #3

- 用try except来发起一个自定义的ValueError的报错
 - 自定义的报错内容为 “This is a custom error message.”
 - 屏幕中打出报错内容

雨课堂投稿

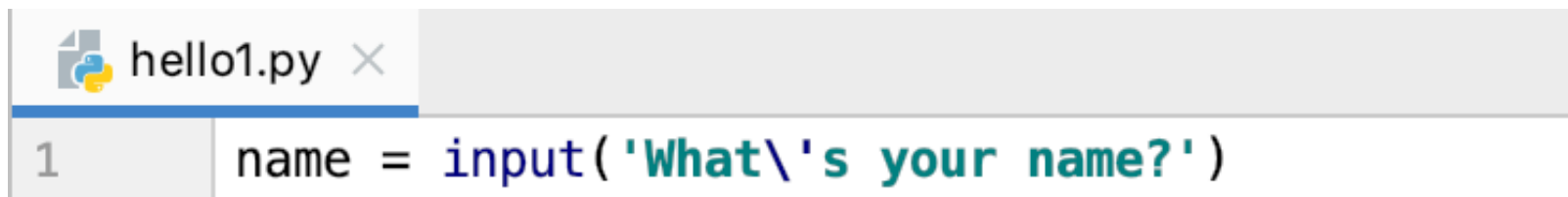
- 登录www.yuketang.cn



变量 Variables

- 赋值 `=`
- 与外界(命令行/终端)交互
 - 允许用户输入信息并储存

`input()`



```
hello1.py ×  
1 name = input('What\'s your name?')
```

字符串 String

- 字符串连接

- 两个字符串连在一起

+

- 一个字符串重复几次

*

- 字符串中某几个字母

string[*index1:index2*]

数字 Numbers

- 用户输入的默认是字符串
 - 将格式调整为整数(integer)类型

`int()`

- 将格式调整为浮点(float)类型

`float()`

练习 #4

- 让用户在终端中输入两个数字
 - 如果用户输入的不是数字，报错（Please input a number）并退出程序
- 计算两个数字的商
- 输出 “The result is xxx”
 - xxx是两个数字计算的结果

函数 Functions

- 一些动作

int()

float()

- 自定义函数

- 函数名
- 输入参数（非必须）
- 输出（非必须）
- 函数内的变量是**局部的**

```
def functionName(argument1, argument2, ...):  
    """ Write Comment Here """  
  
    ... Codes ...  
  
    return value (Optional)
```

```
1 def add(x, y):  
2     """Add number x and y"""  
3     result=x+y  
4     return result
```

函数 Functions

- 对一些动作进行命名
 - 更易于阅读和调试
- 通过消除重复的代码来缩短代码长度
 - 稍后只需在一个地方进行更改
- 可将大程序分割由多人开发，团队分工
 - 编写调试代码的时候可以只关注一个部分

练习 #5

- 制作一个程序，需要用户输入一段文字，然后右对齐
 - 自定义右对齐函数right_justified()
 - 函数作用是将字符串右对齐
 - Word每行可以写72个字符
 - 提示：右对齐即单词最后一个字母在第72个位置
 - `len()` 可以求得字符串的长度
 - 输入：字符串
 - 输出：字符串

包 Library

- 其他人的程序或你自己的程序

- 在终端中安装包

```
pip install libraryName
```

```
pip3 install -r textFileName.txt
```

- 导入包：代码的**重复使用**

```
Import libraryName
```

```
From libraryName import function
```

```
PS C:\Users\DELL\Dropbox> pip install numpy
Requirement already satisfied: numpy in c:\users\dell\appdat
PS C:\Users\DELL\Dropbox> pip3 install -r requirements.txt
```

```
1  #library
2  from sys import argv
3  import example
4
5  print(example.add(int(argv[1]), int(argv[2])))
```

包 Library

- 导入自己的代码
 - `if __name__ == "__main__":` 后面的内容不会执行

练习 #6

- 导入练习#5中写好的right_justified函数
 - 可适当调整练习#5中的代码
- 无需用户输入一段文本
- 在命令行运行python文件时，**直接输入文本**，并进行右对齐

条件 Conditionals

- 程序运行流程控制
- 条件可以套条件

```
if Condition:  
    code  
elif Condition:  
    if Condition:  
        code  
else:  
    code
```

```
13 x = int(input("What's x? "))  
14 y = int(input("What's y? "))  
15  
16 if x < y:  
17     print("x is less than y")  
18 elif x > y:  
19     print("x is greater than y")  
20 else :  
21     print("x is equal to y")
```


条件 Conditionals

- 关系运算符 Relational operators

符号	含义
>	大于
<	小于
>=	大于等于
<=	小于等于
==	等于
!=	不等于

- 逻辑运算符 Logical operator

符号	含义
and	和
or	或
not	非

<pre>>>> x=1 >>> y=2 >>> x>y or x<y True >>> x>y and x<y False</pre>	<pre>>>> x=1 >>> y=1 >>> not (x==1) False >>> not (x!=y) True</pre>
--	---

条件 Conditionals

- if

```
if Variable == Value1:  
    code1  
elif Variable == Value2:  
    code2  
else:  
    code3
```

- match

```
match Variable:  
    case Value1:  
        code1  
    case Value2:  
        code2  
    case _:  
        code3
```

练习 #7

- 本门课有两个班
 - 01: 12: 55-15: 40
 - 02: 18: 00-20: 20
- 用户输入一个时间
 - 格式 #: ## 或 ##: ##
- 输出上课班级(01 or 02)
 - 没有课输出 “No Class”

- 要求：使用下面的代码框架

- `convert`函数:

- 输入：字符串
- 输出：float

- 功能：将时间文本转换成对应的小时
 - 例：“7: 30” -> 7.5

```
def main():  
    ...  
  
def convert(time):  
    ...  
  
if __name__ == "__main__":  
    main()
```

循环 Loops

- 重复执行一段代码

- While 循环

```
while condition:  
    some codes
```

- For 循环

```
for var in variables:  
    some codes
```

```
6      # while loop  
7      i = 1  
8      while i <= 3:  
9          print("meow")  
10     i = i + 1
```

```
13     # for loop  
14     for i in [0, 1, 2]:  
15         print("meow")
```

```
18     for i in range(3):  
19         print(i)  
20         print("meow")
```

循环 Loops

- 中止循环
 - Condition中已经指明循环次数
 - 未指明循环次数 `break`

```
1 while True:
2     line = input('>>>')
3     if line == 'quit()':
4         break
5     print(line)
```

```
PS C:\Users\DELL\Dropbox\CoursePreparation\2023Fall\Teacher\week2_python> python break.py
>>>Simulate python console
Simulate python console
>>>Wow
Wow
>>>quit()
```

练习 #8

- 上海车牌规则

- 车牌字头为 “沪A”, “沪B”

- 假设无需输入 “沪”

- 号码由5位序号组成

- 第一位和最后一位必须是数字

- 第二、三、四位中任意一位且只有一位必须是英文大写字母

- 数字不能全部为 “0”

```
def main():  
    plate = input("Plate: ")  
    if is_valid(plate):  
        print("Valid")  
    else:  
        print("Invalid")  
  
def is_valid(s):  
    '''  
    '''  
  
main()
```

- 提示: <https://docs.python.org/3/library/stdtypes.html#string-methods>

- 试: A0000, A00000, C12A24, A00E00, AC0123, A0e353, A0CD25

列表 Lists

- 储存一系列变量

- 储存的变量可以被改变

- 获取列表中的其中一个元素 `list[index]`

- 循环列表

```
for var in list:  
    some code
```

- 列表的长度 `len()`

```
1  #List  
2  students = ["Harry", "Ron"]  
3  
4  print(students[0])  
5  print(students[1])  
6  
7  students[0]='Harry Potter'
```

```
30 students = ["Harry", "Ron"]  
31  
32 for student in students:  
33     print(student)
```

列表 Lists

- 添加一个列表的元素

```
t.append(x)  
t = t + [x]
```

- 减少一个列表中的元素

```
t.pop(index)  
t.remove(value)  
del t[index]
```

- 更改列表中的某个元素

```
t[index]=value
```

```
>>> t=[3, 'a', 6]  
>>> t=t+['b']  
>>> t  
[3, 'a', 6, 'b']  
>>> t.pop(2)  
6  
>>> t  
[3, 'a', 'b']  
>>> t.remove(3)  
>>> t  
['a', 'b']  
>>> del t[1]  
>>> t  
['a']
```

错误!!!

```
t.append([x])      # WRONG!  
t = t.append(x)    # WRONG!  
t + [x]            # WRONG!  
t = t + x          # WRONG!
```


练习 #9

- 构建如下函数

- nested_sum: 求列表中的列表里的数字之和

- 例: `nested_sum([[1, 2], [3], [4, 5, 6]])==21`

- cum_sum: 给定一个列表，输出到该位置的累计和

- 例: `cum_sum([1, 2, 3]) == [1, 3, 6]`

- middle: 给定一个列表，输出一个新的列表不包含原列表第一个和最后一个元素

- 例: `middle([1, 2, 3]) == [2]`

元组 Tuples

- 储存一系列变量

- 储存的变量不可被改变

```
19 #tuple  
20 students = ("Harry", "Ron", 'Draco')  
21  
22 print(students[0])  
23 print(students[1])
```

- 获取元组中的其中一个元素

`tuple[index]`

- 循环元组

```
for var in tuple:  
    some code
```

- 元组的长度

`len()`

元组 Tuples

- 函数输出多个结果
 - 储存成元组更容易
 - 执行速度比列表快
 - 更安全

```
1 def max_min(t):  
2     return max(t), min(t)
```

```
>>> max_min([1,2,3])  
(3, 1)  
>>> max, min = max_min([1,2,3])  
>>> max  
3  
>>> min  
1
```

集合 Set

- 唯一值的集合

set()

- 集合可以改变

符号	含义
$a - b$	差集：在a不在b中的元素集合
$a \mid b$	并集：在a或b以及ab交集集中的元素
$a \& b$	交集：在a和b中的元素
$a \wedge b$	在a或b但不在ab交集集中的元素

- 添加元素

set.add(value)

- 删除元素

set.remove(value)

```
>>> students=['Harry','Harry','Ron']
...
>>> set(students)
{'Ron', 'Harry'}
```

```
>>> a=set('123452345')
>>> b=set('45678456')
>>> a
{'1', '3', '4', '2', '5'}
>>> b
{'7', '6', '4', '8', '5'}
>>> a - b
{'2', '1', '3'}
>>> a | b
{'1', '7', '6', '3', '4', '8', '2', '5'}
>>> a & b
{'4', '5'}
>>> a ^ b
{'1', '8', '7', '6', '2', '3'}
```

```
>>> a
{'B', 'A', 'C'}
>>> a.add('D')
>>> a
{'D', 'B', 'A', 'C'}
>>> a.remove('C')
>>> a
{'D', 'B', 'A'}
```

字典 Dictionary

- 字典是一种映射

- 将键(key)与值(value)关联 `{key: values}`

- 获取字典中某个键对应的值 `dict[key]`

- 获取字典中所有的键 `dict.keys()`

- 获取字典中所有的值 `dict.values()`

- 在原有字典中添加一个新的内容 `dict[new_key]=new_value`

Name -- Key	House -- Value
Harry Potter	Gryffindor
Ron Weasley	Gryffindor
Draco Malfoy	Slytherin

```
# dictionary
students = {
    "Harry Potter": "Gryffindor",
    "Ron Weasley": "Gryffindor",
    "Draco Malfoy": "Slytherin",
}
print(students["Harry Potter"])
print(students["Ron Weasley"])
print(students["Draco Malfoy"])
```

```
>>> students.values()
dict_values(['Gryffindor', 'Gryffindor', 'Slytherin'])
>>> students.keys()
dict_keys(['Harry Potter', 'Ron Weasley', 'Draco Malfoy'])
```

```
>>> students['Lulu']='Gryffindor'
>>> students
{'Harry Potter': 'Gryffindor', 'Ron Weasley': 'Gryffindor', 'Draco Malfoy': 'Slytherin', 'Lulu': 'Gryffindor'}
```

字典 Dictionary

- 循环获取字典中的内容

```
for key in dict:  
    some code
```

```
for key, value in dict.items():  
    some code
```

```
>>> for key in students:  
...     print(key)  
...  
Harry Potter  
Ron Weasley  
Draco Malfoy
```

```
>>> for key in students:  
...     print(key, students[key])  
...  
Harry Potter Gryffindor  
Ron Weasley Gryffindor  
Draco Malfoy Slytherin
```

```
>>> students.items()  
dict_items([('Harry Potter', 'Gryffindor'), ('Ron Weasley', 'Gryffindor'), ('Draco Malfoy', 'Slytherin')])  
  
>>> for key, value in students.items():  
...     print(key, value)  
...  
Harry Potter Gryffindor  
Ron Weasley Gryffindor  
Draco Malfoy Slytherin
```

字典 Dictionary

- 键可以是字符串、数字或元组

合理

字符串: {'key1': 'value1', 'key2': 3}

数字: {0:0, 2:0}

元组: {(1,1):1, (1,2):1}

- 键不可以是列表或集合

错误

列表: {[1,1]:1, [1,2]:1}

集合: {{1,1}:1, {1,2}:1}

练习 #10

- 构建函数histogram:
 - 输入：一段字符串
 - 输出：每个字母出现的次数

类 Class

- 创建自定义的数据类型
 - 储存自定义的属性
 - 类的名称需要大写
- 类储存的内容是可以更改的
 - 获取自定义类的其中一个属性
 - 修改自定义类的其中一个属性

```
class Name:  
    def __init__(self, arg1,arg2,...):  
        self.attribute1=arg1  
        self.attribute2=arg2  
    .....
```

class.*attribute1*

```
>>> lulu=Student('LuLu','AI')  
>>> lulu.major  
'AI'
```

```
1 class Student:  
2     def __init__(self, name, major):  
3         self.name = name  
4         self.major = major  
5         self.score=dict()
```

class.*attribute1*=*new_value*

```
>>> lulu.major='Finance'  
>>> lulu.major  
'Finance'
```

类 Class

- 自定义可以进行的方法(method)

```
class Name:  
    def functionName(self, argument1, argument2, ...):  
        """ Write Comment Here """  
        ... Codes ...  
        return value (Optional)
```

```
7      def update(self, courseName, courseScore):  
8          self.score[courseName]=courseScore  
9  
10     def averageScore(self):  
11         if len(self.score)==0:  
12             return 'No Score'  
13         else:  
14             return sum(self.score.values())/len(self.score)
```

练习 #11

自定义类银行账户Account

- 有两个属性

- 持有人 owner
- 余额 balance
 - 如无特殊说明，余额为0

- 有两个方法

- 存钱deposit
- 取钱withdraw

- 存钱deposit

- 功能：增加余额
- 输入：存钱的数额
- 输出：“Deposit Accepted”

- 取钱withdraw

- 功能：查看余额，确定能否取钱成功，成功则减少余额
- 输入：取钱的数额
- 输出：“Withdraw Accepted” 或 “Not enough Money”

类 Class

- 运算符重载 Operator overloading
 - 覆盖已有的python操作/运算
 - <https://docs.python.org/3/reference/datamodel.html#special-method-names>

`print()` → `def __str__(self):`
`return f"{self.attribute1} some text {self.attribute2}"`

```
7 def __str__(self):  
8     return f"{self.name}'s major is {self.major}"
```

```
>>> lulu=Student('Lulu','AI')  
>>> print(lulu)  
Lulu's major is AI
```

`>` → `def __gt__(self, other):`
`some code`

```
19 def __gt__(self, other):  
20     if self.averageScore() > other.averageScore():  
21         return True  
22     else:  
23         return False
```

```
>>> lulu=Student('Lulu','AI')  
>>> lulu.update('AI',85)  
>>> lulu.update('Econ',70)  
>>> lulu.averageScore()  
77.5
```

```
>>> Alex=Student('Alex','AI')  
>>> Alex.update('Fina',90)  
>>> Alex.update('Acct',95)  
>>> lulu > Alex  
False
```

练习 #12

- 在练习#11的基础上
 - print 一个人的银行账户时显示出 “xxx’s account balance is xxx”
 - 当两个账户相加时，输出两个账户余额的和

类 Class

- 继承 Inheritance

- 在已有的类的基础上进行修改
 - 父类(parent): 已有的类
 - 子类(child): 修改后的新建的类

```
class ChildClassName(ParentClassName):  
    some code
```

```
26 class Professor(Student):  
27     ...  
  
>>> lulu=Professor('lulu','AI')  
>>> lulu.major  
'AI'
```

```
BaseException  
+-- KeyboardInterrupt  
+-- Exception  
    +-- ArithmeticError  
    |   +-- ZeroDivisionError  
+-- AssertionError  
+-- AttributeError  
+-- EOFError  
+-- ImportError  
    +-- ModuleNotFoundError  
+-- LookupError  
    +-- KeyError  
+-- NameError  
+-- SyntaxError  
    +-- IndentationError  
+-- ValueError  
...  

```

类 Class

- 修改父类中已有的方法

- 在原方法上添加内容

```
super().functionName(arg)
```

```
class Professor(Student):  
    def __init__(self, name, major, title):  
        super().__init__(name, major)  
        self.title = title  
  
>>> lulu = Professor('Lulu', 'AI', 'Assistant Professor')  
>>> lulu.title  
'Assistant Professor'  
>>> lulu.major  
'AI'
```

- 直接覆盖原方法

```
def functionName(self, ...):  
    """ Write Comment Here """  
    ... Codes ...  
    return value (Optional)
```

```
def update(self):  
    self.title = 'Professor'  
  
>>> lulu.update()  
>>> lulu.title  
'Professor'
```

练习 #13

- 在练习#12的基础上
 - 定义一个新的类股票账户StockAccount，他源自于类Account
 - 除了owner, balance之外，他还有一个新属性stock
 - stock是一个字典，键为购买的股票，值为购买的金额
 - StockAccount有一个新方法购买buy
 - 输入：stockName, Amount
 - 功能：余额扣除花费的金额，更新持有人的股票信息
 - print一个人的股票账户时，显示他购买的股票

读写文件 File I/O

- txt

- 写

```
with open("fileName.txt", "a") as file:  
    file.write("some text")
```

```
with open("new_txt.txt", "w") as file:  
    file.write("some text\n")  
    file.write("some text 2\n")
```

- 读

```
with open("fileName.txt", "r") as file:  
    lines = file.readlines()
```

```
with open("new_txt.txt", "r") as file:  
    lines = file.readlines()  
  
>>> lines  
['some text\n', 'some text 2\n']
```

- CSV

- 写

```
import pandas as pd  
dataframe.to_csv("fileName.csv")
```

```
import pandas as pd  
df=pd.DataFrame([{'Name':'Lulu','Major':'AI'},  
                  {'Name':'Alex','Major':'AI'}])  
df.to_csv('some.csv')
```

- 读

```
import pandas as pd  
df=pd.read_csv("fileName.csv")
```

```
df1=pd.read_csv('some.csv')  
>>> df1  
   Unnamed: 0  Name Major  
0            0  Lulu   AI  
1            1  Alex   AI
```

练习 #14

- 代码行 line of code

- 2行

```
# Say hello
```

```
name = input("What's your name? ")  
print(f"hello, {name}")
```

- 5行

```
def is_even(n):  
    if n % 2 == 0:  
        return True  
    else:  
        return False
```

- 制作一个程序，在终端中输入一个txt文件地址，输出该文件中的代码行

- 注释#的行不算代码行
 - 空白的行不算代码行

Homework 0

- 判定英雄技能是否生效

有问题吗？

- 请随时举手提问。

