**《Python 程序设计》课程作业**

**第一次 （2024年2月27日）**

姓名：何卓越

1. **遇到的问题及解决方案**

* **错误或困难描述以及相应的解决方案：**

1. **报错：IndentationError: expected an indented block**

问题：缺少缩进

解决方案：在使用while循环、if语句、函数定义时常见。仔细检查一下就行了。

1. **报错：ValueError: invalid literal for int() with base 10: '17岁'**

问题：尝试将一个非数字字符串转换为整数时引起的

代码详情：

age = int(input("你的年龄是："))

if age >= 18:

print("恭喜！你成年了。")

else:

diff = str(18 - age)

print("要年满18岁才成年，你还差 " + diff + " 岁。")

这里，如果输入为纯数字，那么一切正常，但是当我下意识地加了个“岁”，就出现了这个问题。

解决方案： 将输入的内容中的数字提取出来赋值给age

代码修改详情：

age = input("你的年龄是：")

age = ''.join(filter(str.isdigit,age)) # filter函数会在后续解释

age = int(age)

if age >= 18:

print("恭喜！你成年了。")

else:

diff = str(18 - age)

print("要年满18岁才成年，你还差 " + diff + " 岁。")

1. **总结与思考**

* **技术及思想方面的收获：**

在解决问题的过程中看到了不少函数的用法和额外的知识，以下是这次作业的技术总结：

（1）String模块ascii\_letters和digits

简单介绍一下String模板的 ascii\_letters 和 digits 方法，其中ascii\_letters是生成所有字母，从a-z 和A-Z，digit是生成数字0-9

示例如下：

In [1]: chars = string.ascii\_letters + string.digits

print(chars)

Out[1]: abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789

（2）format函数的用法

1.format说明

format作为Python的格式化字符串函数，主要通过字符串中的花括号{}来识别替换字符，从而完成字符串的格式化。

In[2]: print("我叫{}，今年{}岁。".format("小帅",18))

#花括号的个数决定了参数的个数。但是花括号的个数可以少于参数。

print("我喜欢{}和{}".format("乒乓球","羽毛球","敲代码"))

#花括号多于参数的个数，则会报错。

Out[2]: 我叫小帅，今年18岁。

我喜欢乒乓球和羽毛球

2.通过数字参数传入位置参数（这也是较为常见的形式）

* 数字必须是大于0的整数
* 带数字的替换字段可以重复
* 数字形式的简单字段名相当于把字段当成一个序列形式。通过索引的形式进行一一取值

In[3]: x=3.1415

y=9.9999

print("{0:.2f} {1:.2f}".format(x,y))

print("{1:.2f} {1:.2f}".format(x,y))

#这里的.2f就是我们常见的保留小数点后两位的操作，还有其他很多种格式**（如 下表格）**

Out[3]: 3.14 10.00

10.00 10.00

| **数字** | **格式** | **输出** | **描述** |
| --- | --- | --- | --- |
| 3.1415926 | {:.2f} | 3.14 | 保留小数点后两位 |
| 3.1415926 | {:+.2f} | +3.14 | 带符号保留小数点后两位 |
| -1 | {:+.2f} | -1.00 | 带符号保留小数点后两位 |
| 2.5645 | {:.0f} | 2 | 不带小数 |
| 6 | {:0>2d} | 06 | 数字补零 (填充左边, 宽度为2) |
| 6 | {:x<4d} | 6xxx | 数字补x (填充右边, 宽度为4) |
| 20 | {:x<4d} | 20xx | 数字补x (填充右边, 宽度为4) |
| 1000000 | {:,} | 1,000,000 | 以逗号分隔的数字格式 |
| 0.75 | {:.2%} | 75.00% | 百分比格式 |
| 1000000000 | {:.2e} | 1.00e+09 | 指数计数法 |
| 100 | {:>10d} | 100靠右对齐 | 右对齐 (默认, 宽度为10 |
| 100 | {:<10d} | 100靠左对齐 | 左对齐 (宽度为10) |
| 100 | {:^10d} | 100居中对齐 | 中间对齐 (宽度为10) |

（3）random函数的用法

random常用的函数:

**random.random()** 返回0-1之间的随机浮点数

In[4]: import random

random.random()

Out[4]: 0.25818427858230764

**random.randint(a,b)** 返回一个a-b之间的随机整数（包括a，b）

**random.choice(sequence)**  从sequence中返回一个随机数，sequence可以是列表、元组或字符串，但不能为空。

In[5]: import random

print(random.choice("学习 python"))

print(random.choice(["JGood", [0], "is", "a", [0], "handsome", "boy"]))

print(random.choice(("Tuple", [0], "List", "Dict")))

Out[5]: y

JGood

[0]

**random.shuffle(X[,random])** 用于将列表中的元素打乱顺序，俗称“洗牌”。

In[6]: import random

demo\_list = ["python", "is", "powerful", "simpel", "and so on..."]

random.shuffle(demo\_list)

print(demo\_list)

Out[6]: ['and so on...', 'is', 'python', 'powerful', 'simpel']

**random.sample(sequence,K)**  从指定序列中随机获取K个元素作为一个片段返回

In[7]: import random

list\_num =[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

# 定义temp为碎随机在list\_num中获取的3个元素

temp = random.sample(list\_num, 3)

print(temp)

# 原有的序列没有发生改变

print(list\_num)

Out[7]: [6, 10, 5]

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

（4）join函数的用法

**作用**：将序列中的元素以指定的字符连接生成一个新的字符串。

**语法**：‘sep’.join(seq)

**参数说明**：sep：分隔符。可以为空（也很常用）

seq：要连接的元素序列（列表、字符串、元组、字典）

**返回值**：连接后生成的新字符串。

In[8]: a="abcd"

print(",".join(a))

print("|".join(['a','b','c']))

print(",".join(('a','b','c')))

Out[8]: a,b,c,d

a|b|c

a,b,c

（5）filter函数的用法

**功能：**

filter() 函数用于 过滤 可迭代对象中不符合条件的元素，返回由符合条件的元素组成的新的迭代器。filter() 函数把传入的函数依次作用于每个元素，然后根据返回值是 True 还是 False，来决定保留或丢弃该元素。

**参数说明：**

(1) function：用于实现判断的函数，可以为 None。

(2) iterable：可迭代对象，如列表、range 对象等。

(3) 返回值：返回一个迭代器对象。

**filter() 函数的基本应用：**

a.使用 filter() 函数过滤出 0~100 (不包括 100) 之间的所有奇数，代码如下：

In[9]: list(filter(lambda x:x&1, range(100)))

Out[9]: [1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, 35, 37, 39, 41, 43, 45, 47, 49, 51, 53, 55, 57, 59, 61, 63, 65, 67, 69, 71, 73, 75, 77, 79, 81, 83, 85, 87, 89, 91, 93, 95, 97, 99]

b.筛选指定范围的数据

定义学生列表，对 2023 年高考理科成绩进行筛选，获得成绩在 500 分到 600 分之间的学生信息，代码如下：

In[10]: list1 = [('小明', 500), ('小刚', 502), ('小王', 700), ('小李', 400)]

Out[10]: print(list(filter(lambda x: 500 <= x[1] <= 600, list1)))

[('小明', 500), ('小刚', 502)]

c. 筛选类型不是字符的元素

在一个集合中有一组数据，包括字符串、数字等，筛选出类型不是字符串的元素，代码如下：

In[11]: list1 = ['小明', 18, 176, 85]

print(list(filter(lambda x:type(x)!=str, list1)))

Out[11]: [18, 176, 85]

d.取出字符串中的数字并转换为整型

取出一组注册信息中的数字，然后转换为整数。首先判断字符串是否为数字，然后使用 filter() 函数过滤数字，最后转换为整型输出，代码如下：

In[12]: s = 'abds123sadoi456'

print("".join(list(filter(str.isdigit, s))))

Out[12]: 123456

e. 去除序列中所有值为假的元素

如果将 filter() 函数的第一个参数 function 的值设置为 None，就会默认去除序列中所有值为假的元素，如 None、False、0、’’、()、[] 和 {}等，代码如下：

In[13]: list1 = ['', False, 1, 0, None, [], 3, 4, [1,23]]

print(list(filter(None, list1)))

Out[13]: [1, 3, 4, [1, 23]]

f. 获取索引中以索引为基数所对应的元素

通过 filter() 和 lambda() 函数输出列表 list\_a 中以索引为基数出现次数最多的元素。代码如下：

In[14]: list1 = [12, 213, 22, 2, 2, 2, 22, 2, 2, 32]

print(list(filter(lambda x: x in list1, [i for i in range(len(list1))])))

Out[14]: [2]

g.其中，提到的lambda表达式，又称[匿名函数](https://so.csdn.net/so/search?q=%E5%8C%BF%E5%90%8D%E5%87%BD%E6%95%B0&spm=1001.2101.3001.7020" \t "https://blog.csdn.net/qq_55858843/article/details/_blank)，常用来表示内部仅包含 1 行表达式的函数。如果一个函数的函数体仅有 1 行表达式，可以用 lambda 表达式来代替。