1.3 Introduction-Sequence Mutation变化

- sequence: 有order的collection
  - string
    - 用三引号当作可以跨行的string
    - s=""中间如果有空格的话就是1个character 没有的话就是0
    - s=“5”和s=5是不一样的东西
  - list 包含了很多data values
    - 用["spam", "bungee", "swallow"]去create
    - 易错
      - What is the type of m?
      - l = ['w', '7', 0, 9]m = l[1:2]
      - 答案是 list， a slice returns a list no matter how large the slice
      - l = ['w', '7', 0, 9] m = l[1] 但是这个就是string 上一个是slicing 这个是accessing an element
  - tuple
    - 跟list差不多但就是用()
    - julia = ("Julia", "Roberts", 1967, "Duplicity", 2009, "Actress", "Atlanta, Georgia")
    - 和list最大的区别就是 tuple是不可更改的
    - 看区别
      - t = (5,) print(type(t))
      - x = (5) print(type(x))
      - 输出分别是<class 'tuple'> <class 'int'> 所以如果要tuple必须在最后加一个逗号 对于整数来说 因为python会怀疑你到底要干啥
    - 一个程序
      - s="Python"
      - myList=["one",2,"three"]
      - print(myList[1])
      - 建了一个tuple叫做myList 但是想要其中第二项的话要选1 因为第一项是从0开始的
      - print(s[len(s)-1]) 或者print(s[-1])
      - 也可以最后一项是-1然后往前就是-2 -3
      - sports = ['cricket', 'football', 'volleyball', 'baseball', 'softball', 'track and field', 'curling', 'ping pong', 'hockey']

        - last=sports[-3:] 因为冒号后面是不包括的 所以如果要到最后一个就不要输入数字
    - len 长度函数
    - []的用法
        - create list & indexing 索引
            - new_lst = ["NFLX", "AMZN", "GOOGL", "DIS", "XOM"]
            - part_of_new_lst = new_lst[0]
        - 区别
            - lst = [0]
            - n_lst = lst[0]
            - print(lst)
            - print(n_lst)
            - 输出是 [0] 和 0
        - 一个分别
            - alist = ["hello", 2.0, 5]
            - print(len(alist))
            - print(len(alist[0]))
            - 输出一个是3 是说alist这个list有三个内容 一个是5 是alist的第一个内容“hello”有五个字母
        - s = "python rocks" print(len(s)) 答案是12 因为空格也算一个character
        - L = [0.34, '6', 'SI106', 'Python', -2] print(len(L[1:-1])) 答案是3个 从第二项算起 但是不包括-1这项 其实[1:-1]就看间隔 从第一个间隔到从右边往左数第一个间隔 不包括最后一个的内容
        - 例子
            - new_lst = ["computer", "luxurious", "basket", "crime", 0, 2.49, "institution", "slice", "sun", ["water", "air", "fire", "earth"], "games", 2.7, "code", "java", ["birthday", "celebration", 1817, "party", "cake", 5], "rain", "thunderstorm", "top down"]
            - sub_lst= new_lst[8:12] 为啥是8:12

- concatenation and repetition
    - print([1,2]+[3,4]) 就是创造了一个list 是[1,2,3,4]
    - print([0]*4) 变成 [0,0,0,0]
- count：数list里到底出现了几次
    - sequence name. method name 比如
    - a = "I have had an apple on my desk before!" print(a.count("e")) 显示5
    - z = ['atoms', 4, 'neutron', 6, 'proton', 4, 'electron', 4, 'electron', 'atoms'] print(z.count("a")) 显示0

    - print(z.count("4")) 显示0 因为是string格式不是integer格式
- index:找到item在哪里 定位
    - music = "Pull out your music and dancing can begin" print(music.index("m"))显示 14 print(music.index("your"))显示9 空格数到这个的前面的数字
    - qu = "wow, welcome week!" ty = qu.index("we") 答案是5 不管出现了几次 index只会看最开始出现的那次的定位
- split：把每个list里的字符或者单词之间分开组建成sub list
    - 定位空格split
        - song = "The rain in Spain..."
        - wds = song.split() 定位空格的时候默认（）就空着就行
        - print(wds)
        - 输出 ['The', 'rain', 'in', 'Spain...']
    - 也可以定位某个字母删除
        - song = "The rain in Spain..."
        - wds = song.split('ai')
        - print(wds)
        - 把ai全删了 然后分开 ['The r', 'n in Sp', 'n...']
- join 把几个合起来
    - wds = ["red", "blue", "green"]
    - glue = ';'
    - s = glue.join(wds)
    - print(s)
    - print(wds)
    - print("***".join(wds))
    - print("".join(wds))
    - 输出
    - red;blue;green
    - ['red', 'blue', 'green']
    - red***blue***green
    - redbluegreen
- for loop循环
    - 一个例子
        - for name in ["Joe", "Amy", "Brad", "Angelina", "Zuki", "Thandi", "Paris"]:
        - print("Hi", name, "Please come to my party on Saturday!")
        - 输出Hi Joe Please come to my party on Saturday!

    - Hi Amy Please come to my party on Saturday!
    - Hi Brad Please come to my party on Saturday!
    - Hi Angelina Please come to my party on Saturday!
    - Hi Zuki Please come to my party on Saturday!
    - Hi Thandi Please come to my party on Saturday!
    - Hi Paris Please come to my party on Saturday!
- 另一个例子
    - for achar in "Go Spot Go":
    - print(achar)
    - 打出来的结果就是竖着 包含空格
- for loop的取名方法
    - 最好是 for fruit in fruits这种 前面是单数 后面是复数的形式 不会confuse
- printing intermediate results
    - 案例
        - 最开始想算tot
        - w = range(10)
        - tot = 0
        - for num in w:
        - 　　tot += num 这里翻译过来就是tot=num+tot
        - print(tot)
        - 但是想看中间过程 最开始的改进是这样的
        - w = range(10)
        - tot = 0
        - for num in w:
        - 　　print(num)
        - 　　tot += num
        - print(tot)
        - 输出0 1 2 3 4 5 6 7 8 9 45
        - 但也想知道tot发生了什么 所以 改进代码
        - w = range(10)
        - tot = 0
        - for num in w:
        - 　　print(num)
        - 　　tot += num
        - 　　print(tot)

- print(tot)
- 但是这样又看得不是很懂 又改进
- w = range(10)
- tot = 0
- print("***** Before the For Loop ******")
- for num in w:
- print("***** A New Loop Iteration ******")
- print("Value of num:", num)
- tot += num
- print("Value of tot:", tot)
- print("***** End of For Loop *****")
- print("Final total:", tot)
- 结果就变成

```
***** Before the For Loop ******
***** A New Loop Iteration ******
Value of num: 0
Value of tot: 0
***** A New Loop Iteration ******
Value of num: 1
Value of tot: 1
***** A New Loop Iteration ******
Value of num: 2
Value of tot: 3
***** A New Loop Iteration ******
Value of num: 3
Value of tot: 6
***** A New Loop Iteration ******
Value of num: 4
Value of tot: 10
***** A New Loop Iteration ******
Value of num: 5
Value of tot: 15
***** A New Loop Iteration ******
Value of num: 6
Value of tot: 21
***** A New Loop Iteration ******
Value of num: 7
Value of tot: 28
***** A New Loop Iteration ******
Value of num: 8
Value of tot: 36
***** A New Loop Iteration ******
Value of num: 9
Value of tot: 45
***** End of For Loop *****
Final total: 45
```

-

- Accumulate patern
  - accumulate variable 累计变量
  - iterator variable 指示变量
    - iterator variable 和 iterable的区别
      - iterable就是在for loop里面 for.. in...in后面的东西 是sequence 是the thing that you iterate with
      - 这个东西=[]就是list; =()可能就是tuple =""可能就是string
      - 需要注意的是 如果=某个数字的话 其实是error的 因为integer不是sequence
      -
  - update accumulator
  - 看个例子
    - nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
    - 上面那个其实也可以写作 nums=range(1,11)
    - range 方程
      - range(5)就是range[0,1,2,3,4]
      - 这就是创造了一个list了
    - accum = 0 最原始的累计变量设为0
    - for w in nums: w就是iterator variable指示变量
    - accum = accum + w
    - print(accum) 答案是10
  - 相似案例
    - accum = 0
    - for w in range(11):
    - accum = accum + w
    - print(accum)
    - 这样输出才是55 要把accum放在for loop外面
  - 还有个例子
    - print(list(range(5)))
    - print(list(range(0,5)))
    - 如果要把弄出来的东西转化成list就还要用list function 输出 [0, 1, 2, 3, 4] 转化成list是为了在iteration迭代之外也能用
  - 另一个折磨人的例子！ count句子里的字符数without len()
    - str1 = "I like nonsense, it wakes up the brain cells. Fantasy is a necessary ingredient in living."
    - accum=0 如果想要累加 accum一定要放外面

    - for ch in str1: 对每个str1中的字符来说
    - ch=1 设定每个字符为1
    - accum=accum+ch accum每次遇到一个字符就更新一次
    - numbs=accum 最后的数量assign to numbs
- 再一个超级超级恶心想了很久的例子 Write code to create a list of word lengths for the words in original_str using the accumulation pattern and assign the answer to a variable num_words_list. (You should use the len function).
    - original_str = "The quick brown rhino jumped over the extremely lazy fox"
    - words=original_str.split() 先把句子split变成每个单词的分开的list
    - num_words_list=[] 先在forloop之前创建一个空list
    - for word in words: 对于分开的words list里的单词
    - num_words_list=num_words_list+[len(word)] 把每个单词的长度加进list里
- 组合拳
    - fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
    - for n in range(5):
    - print(n, fruits[n])
    - 输出结果是
    - 0 apple
    - 1 pear
    - 2 apricot
    - 3 cherry
    - 4 peach
    - 或者输入还可以
    - fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
    - for n in range(len(fruits)):
    - print(n, fruits[n])
- image
    - A digital image is a finite collection of small, discrete picture elements called pixels
    -

Method Name	Example	Explanation
Pixel(r,g,b)	Pixel(20,100,50)	Create a new pixel with 20 red, 100 green, and 50 blue.
getRed()	r = p.getRed()	Return the red component intensity.
getGreen()	r = p.getGreen()	Return the green component intensity.
getBlue()	r = p.getBlue()	Return the blue component intensity.
setRed()	p.setRed(100)	Set the red component intensity to 100.
setGreen()	p.setGreen(45)	Set the green component intensity to 45.
setBlue()	p.setBlue(156)	Set the blue component intensity to 156.

-

Method Name	Example	Explanation
Image(filename)	img = image.Image("cy.png")	Create an Image object from the file cy.png.
EmptyImage()	img = image.EmptyImage(100,200)	Create an Image object that has all "White" pixels
getWidth()	w = img.getWidth()	Return the width of the image in pixels.
getHeight()	h = img.getHeight()	Return the height of the image in pixels.
getPixel(col,row)	p = img.getPixel(35,86)	Return the pixel at column 35, row 86.
setPixel(col,row,p)	img.setPixel(100,50,mp)	Set the pixel at column 100, row 50 to be mp.

- 一张图片看长宽和色彩强度
  - import image
  - img = image.Image("luther.jpg")
  - print(img.getWidth())
  - print(img.getHeight())
  - p = img.getPixel(45, 55)
  - print(p.getRed(), p.getGreen(), p.getBlue())
  - 输出 这里是看某个位点的色彩强度 用坐标轴
  - 400

    - 244
    - 165 161 158
- 外部迭代和内部迭代
    - for i in range(5): for j in range(3): print(i, j)
    - for row in range(img.getHeight()): for col in range(img.getWidth()):
    - 对每个像素迭代
    - negative image的意思：比如R 红色的255 某个component是50 那opposite or negative就是255-50=205 每个pixel都这样之后就有点像X光片弄出来的感觉了 还可以调灰..有点像 个每个像素都加了一层滤镜