Ficha 3 - Estruturas de controlo resolvida

Jorge Gustavo Rocha

May 30, 2020

1 Estruturas de Controlo em Python

Jorge Gustavo Rocha
Departamento de Informática, Universidade do Minho
27 de abril de 2020

Um programa geralmente não segue uma execução linear de todas as instruções.

Há instruções que só que querem executar em determinadas condições. Para tal, usa-se uma estrutura de controlo:

`if..else...`

Noutros casos, pretende-se repetir múltiplas vezes o mesmo conjunto de instruções. Para tal, usam-se as estruturas de controlo:

`while`
`for`

2 Dados

Para estes exercícios com estruturas de controlo, vamos usar caixas para entrada de dados (usando a função `input()` e as estruturas de dados seguintes.

[1]:
```
import numpy as np
populacao = { "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras␣
 ↪de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe":␣
 ↪53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro":␣
 ↪7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila␣
 ↪Verde": 49171, "Vizela": 24477 }
vel = np.array([ 50, 50, 70, 90, 120 ])
```

2.1 Entrada de dados

1. (**Resolvido**) Leia o nome do utilizador e apresente o nome em maiúsculas.

[2]:
```
nome=input()
```

```
Gustavo
```

[3]:
```
nome.upper()
```

[3]:
```
'GUSTAVO'
```

2. Leia o ano de nascimento e diga quantos anos o utilizador tem.

[4]:
```
from datetime import date
ano_atual = date.today().year

ano_nascimento = input()
print("Você tem {} anos.".format(ano_atual - int(ano_nascimento) ))
```

```
1969
Você tem 51 anos.
```

3. Leia uma palavra e mostre-a de trás para frente. Use as sugestões do stackoverflow para calcular a string ao contrário.

[5]:
```
palavra = input()
print(palavra[::-1])
```

```
O céu está azul
luza átse uéc O
```

2.2 Estruturas condicionais

1. Leia um número de diga se é um número inteiro.
2. Leia um número de diga se é um número real.
3. Leia um número de diga se é positivo, negativo ou zero.
4. Leia um número e diga se é par ou ímpar, mas se e só se o número for inteiro. Se não for um inteiro, diga: "Número inválido: tem que ser um número inteiro"
5. Leia a data de nascimento e diga a idade que o utilizador tem. Use a função `date.today()` para saber a data de hoje (tem que preceder com `from datetime import date`).
6. Leia o nome próprio do utilizador. Responda 'válido', se o nome não contém nenhum espaço. Resposta 'inválido', se o nome contém um espaço como em `Ana Rita`, por exemplo.

2.3 Estruturas condicionais

1. Leia um número de diga se é um número inteiro.

Quando se lê algo fornecido pelo utilizador, ficamos sempre dependentes do que for introduzido pelo utilizador. Nunca é garantido que o utilizador preenche os dados pedidos da melhor forma. Por isso, quando se pede uma data de nascimento, por exemplo, o utilizador pode por '1975', pode por só '05' ou pode por 'aaaa', o que nem sequer é um número.

A entrada é sempre uma string.

Para responder a esta questão, há duas estratégias possíveis: - Tentar converter a entrada (a string) para int e ver se corre bem ou mal, - Ver se a string é composta apenas por algarismos.

Vamos resolver o exercício de duas maneiras diferentes, seguindo estas duas abordagens.

2.3.1 1. Resposta (converter para `int` e ver se corre bem)

[6]:
```
x = input()
try:
    numero = int(x)
except ValueError:
    print("A entrada {} não é um inteiro".format(x))
else:
    print("O número {} é um inteiro".format(numero))
```

```
12
O número 12 é um inteiro
```

Nota: quando se usa um bloco `try..except` e ocorre uma exceção, o kernel Python que estás por trás a correr o código Python bloquea. Ou seja, deixa de poder correr o código que está nas células, porque o kernel está parado.

Para se poder continuar a correr o código das células normalmente, sem que o kernel fique bloqueado quando é gerada uma exceção, é preciso marcar a célula com um tag especial raises-exception. Tem que ir a View → Cell Toolbar → Tags. Passa a ver uma nova barra por cima das células. Acrescente a tag raises-exception nas células onde usa blocos `try..except`, **como acontece neste notebook**.

Depois pode voltar a esconder a toolbar das célunas, indo a View → Cell Toolbar → None.

2.3.2 1. Resposta alternativa (verificar se a entrada só contém algarismos)

Neste caso usa-se o módulo de expressões regulares `re` e testa-se com o método `re.match()` se a entrada é formada só por algarismos `\d`, que se repetem 1 ou mais vezes `+`.

[7]:
```
import re
x = input()
if re.match('\d+', x):
    numero = int(x)
    print("O número {} é um inteiro".format(numero))
else:
    print("A entrada {} não é um inteiro".format(x))
```

```
   12
A entrada    12 não é um inteiro
```

Note que as duas soluções não são exatamente iguais. Na primeira, pode escrever '___12__' e, mesmo com os espaços antes e depois, a entrada é convertida para `int`. Na segunda solução, a expressão regular só permite algarismos. Verifique essa diferença entre as duas soluções.

Pode-se melhorar a expressão regular para permitir carateres brancos (espaços ou tabs) antes ou depois dos algarismos, com:

```
if re.match('\s*\d+\s*', x):
```

2. Leia um número de diga se é um número real.

[8]:
```
x = input()
try:
    numero = float(x)
except ValueError:
    print("A entrada {} não é um número real".format(x))
else:
    print("O número {} é um real".format(numero))
```

```
3.1415927
O número 3.1415927 é um real
```

3. Leia um número de diga se é positivo, negativo ou zero.

[9]:
```
x = input()
try:
    numero = float(x)
except ValueError:
    print("A entrada {} não é um número válido".format(x))
else:
    if numero == 0:
        print("O número introduzido é zero")
    elif numero < 0:
        print("O número introduzido é negativo")
    else:
        print("O número introduzido é positivo")
```

```
0.000001
O número introduzido é positivo
```

1. Leia um número e diga se é par ou ímpar, mas se e só se o número for inteiro. Se não for um inteiro, diga: "Número inválido: tem que ser um número inteiro"

[10]:
```
x = input()
try:
    numero = int(x)
except ValueError:
    print("Número inválido: tem que ser um número inteiro")
else:
    quociente, resto = divmod(numero, 2)
    if resto == 0:
        print("O número {} é um par".format(numero))
    else:
        print("O número {} é um ímpar".format(numero))
```

```
12.0
Número inválido: tem que ser um número inteiro
```

4. Leia a data de nascimento e diga a idade que o utilizador tem. Use a função `date.today()` para saber a data de hoje (tem que preceder com `from datetime import date`).

Nota: nesta solução, usa-se a comparação de pares no `if`.

[11]:
```
from datetime import date, datetime
today = date.today()

data_nascimento = input()
try:
    nascimento = datetime.strptime(data_nascimento, "%Y-%m-%d").date()
except ValueError:
    print("Data inválida. Escreva uma data na forma: AAAA-MM-DD")
else:
    anos = today.year - nascimento.year
    # a idade exacta depedende se já celebrou o aniversário este ano
    if ((today.month, today.day) < (nascimento.month, nascimento.day)):
        anos = anos - 1
    print("Tem {} anos.".format(anos))
```

```
1969-07-25
Tem 50 anos.
```

5. Leia o nome próprio do utilizador. Responda 'válido', se o nome não contém nenhum espaço. Resposta 'inválido', se o nome contém um espaço como em `Ana Rita`, por exemplo.

[12]:
```
nome = input()
if nome.find(' ') == -1:
    print("Válido")
else:
    print("Inválido")
```

```
Maria da Fonte
Inválido
```

2.4 Estruturas cíclicas: `for`

1. Use um ciclo `for` para mostrar o nome dos concelhos do dicionário `populacao`.

[13]:
```
for c in populacao:
    print(c)
```

```
Amares
Barcelos
Braga
Cabeceiras de Basto
Celorico de Basto
Esposende
Fafe
Guimarães
Póvoa de Lanhoso
Terras de Bouro
```

```
Vieira do Minho
Vila Nova de Famalicão
Vila Verde
Vizela
```

2. Use o mesmo ciclo `for` e mostre apenas os concelhos do dicionário `populacao` que têm mais de 50 000 habitantes

[14]:
```
for c in populacao:
    if (populacao[c] > 50000):
        print(c)
```

```
Barcelos
Braga
Fafe
Guimarães
Vila Nova de Famalicão
```

3. Use um ciclo `for` para calcular a média das velocidades do vetor `vel`.

[15]:
```
soma = 0
for v in vel:
    soma = soma + v
print(soma/vel.size)
```

```
76.0
```

[16]:
```
# Em alternativa, pode-se (e deve-se) usar:
vel.mean()
```

[16]: 76.0

4. Use um ciclo `for` para calcular a velocidade máxima que consta do vetor `vel`.

[17]:
```
maximo = 0
for v in vel:
    if v > maximo:
        maximo = v
print(maximo)
```

```
120
```

[18]:
```
# Em alternativa, Pode-se (e deve-se) usar:
vel.max()
```

[18]: 120

2.5 Estruturas cíclicas: `while`

1. Use um ciclo `while` para percorrer o dicionário `populacao` e mostrar os concelhos que seriam precisos para juntar no mínimo 200 000 habitantes.

[19]:
```
populacao = { "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras␣
 ↪de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe":␣
 ↪53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro":␣
 ↪7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila␣
 ↪Verde": 49171, "Vizela": 24477 }

soma = 0
concelhos = []
# vamos removendo os concelhos do dicionário à medida que vamos somando a␣
 ↪população
while populacao and soma < 200000:
    x = next(x for x in populacao)
    soma = soma + populacao[x]
    concelhos.append(x)
    print(','.join(concelhos), soma)
    populacao.pop(x)
```

```
Amares 19853
Amares,Barcelos 144408
Amares,Barcelos,Braga 320562
```

[20]:
```
# Com um ciclo for, usando o break para sair do ciclo
# Abordagem mais simples do que usando um ciclo while
# Não destroi o dicionário original populacao
populacao = { "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras␣
 ↪de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe":␣
 ↪53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro":␣
 ↪7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila␣
 ↪Verde": 49171, "Vizela": 24477 }

soma = 0
for x in populacao:
    soma = soma + populacao[x]
    if soma > 200000:
        break
print(soma)
```

```
320562
```

2. Use um ciclo `while` para percorrer o dicionário `populacao` e mostrar os três primeiros concelhos que tenham o nome formado por mais do que uma palavra (como `"Cabeceiras de Basto"`, por exemplo.

[21]:
```
populacao = { "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras␣
↪de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe":␣
↪53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro":␣
↪7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila␣
↪Verde": 49171, "Vizela": 24477 }

concelhos = []
# vamos removendo os concelhos do dicionário que vão sendo testados
while populacao and len(concelhos) < 3:
    x = next(x for x in populacao)
    if (x.find(' ') != -1):
        concelhos.append(x)
    populacao.pop(x)
print(','.join(concelhos))
```

```
Cabeceiras de Basto,Celorico de Basto,Póvoa de Lanhoso
```

[22]:
```
# Com um ciclo for, usando o break para sair do ciclo
# Abordagem mais simples do que usando um ciclo while
# Não destroi o dicionário original populacao
populacao = { "Amares": 19853, "Barcelos": 124555, "Braga": 176154, "Cabeceiras␣
↪de Basto": 17635, "Celorico de Basto": 19767, "Esposende": 35552, "Fafe":␣
↪53600, "Guimarães": 162636, "Póvoa de Lanhoso": 24230, "Terras de Bouro":␣
↪7506, "Vieira do Minho": 14077, "Vila Nova de Famalicão": 134969, "Vila␣
↪Verde": 49171, "Vizela": 24477 }

concelhos = []
for x in populacao:
    if (x.find(' ') != -1):
        concelhos.append(x)
    if len(concelhos) == 3:
        break
print(','.join(concelhos))
```

```
Cabeceiras de Basto,Celorico de Basto,Póvoa de Lanhoso
```