Tuplas e dicionários

Registros

- São variáveis **compostas heterogêneas**.
- Visam representar um conjunto de variáveis (potencialmente) de tipos diferentes e que estão relacionados.
    - Exemplo: endereço, formado por sub-componentes rua, número, bairro, cidade, etc.
- Em muitas linguagens, servem também como artifício para "burlar" a limitação de só retornar um único resultado em funções (ou métodos).

> 💡 Python *não* possui um tipo básico para registro!

Registros e Tuplas

- A estrutura de Python que parece mais indicada para representar registra é a tupla — tipo ***tuple***.
    - A principal diferença é que os registros suportados em outras linguagens utilizam um nome para cada um dos componentes, enquanto que no tipo tuple o agrupamento é feito somente pela ordem...
        - Neste sentido, Python permite a manipulação de tuplas usando índices e slices, com exatamente as mesmas regras existentes para strings e listas.
        - OBS: Um artifício que pode ser adotado para ter nomes e ficar mais parecido com as outras LP é usar variáveis auxiliares na manipulação/modificação de tuplas...
- Outra diferença é que, como no caso dos strings, o tipo tuple lida com objetos **imutáveis;**

    - Porém, de forma semelhante à manipulação de strings, isto não impede que seja criado outro objeto com um conteúdo diferente (caso seja necessário) e que este novo objeto seja atribuído à mesma variável...
- **Tuplas são delimitadas por parênteses e os elementos são separados por vírgulas;**
- Assim como o tipo list, tuple é um tipo como outro qualquer e pode ser usado sem definição prévia;
- Tuplas também são implementadas como objetos.
- Como no caso das listas, os elementos de uma tupla podem ser atribuídos a variáveis separadas, usando a atribuição múltipla.
    - Da mesma forma, o número de variáveis utilizado deve ser exatamente igual ao tamanho da tupla.
- Existem vários outros operadores e funções para manipular tuplas, geralmente com finalidades ou funcionalidades já cobertas por outros tipos.

```
rua = ‘Rua da Hora’ # Variáveis independentes...
numero = 230
cidade = ‘Recife’
uf = ‘PE’
cep = ‘52020-000’
fones = [12345678, 987654321]
end = (rua, numero, cidade, uf, cep, fones) # Cria a tupla...
endReduz = end [0:2] # Resultado é (‘Rua da Hora’, 230)
endR2 = end[0] + ‘, ’ + str(end[1]) # Idem, em formato string..
r, n, ci, u, ce, f = end # Recuperando todos os itens da tupla.
end [1] = 200 # Causa erro fatal!
```

Tabelas

- Uma aplicação bastante comum de registros são as chamadas **tabelas**:
    - Normalmente, tabelas são implementadas como **um array de registros;**

    - Geralmente, um dos componentes do registro serve como identificador único, o que é chamado de **chave** e normalmente tem tipo numérico (por eficiência).
- Exemplos: Tabela de cursos de uma universidade → A chave será um código de curso

Tipo *dictionary*

- A tabela será um dicionário e os seus registros serão os elementos (itens) do dicionário:
    - A chave do dicionário será a chave da tabela;
    - O conteúdo associado à chave do dicionário (em geral) será uma tupla contendo os outros dados do registro.
- Esta representação é boa quando:
    - a tabela é grande e/ou
    - pode ser alterada durante a execução do programa e/ou
    - não existe uma preocupação com a ordem de impressão dos elementos da tabela.
- OBS: Os dicionários também são implementados como objetos e são **mutáveis**, como as listas.

Sintaxe

- **Dicionários são delimitados por chaves {} com seus elementos separados por vírgulas;**
    - Cada elementos tem uma chave e um conteúdo, que são separados por dois pontos :
    - *Apesar das chaves de um dicionário poderem ter tipo string (e outros), o ideal é que tenham tipo inteiro*
        - Não pode haver duas chaves iguais em um dicionário
- O acesso a um item específico é feito usando a chave escrita entre colchetes [].

```
d1 = { } # Cria um dicionário vazio – mais simples...
d2 = dict () # Função que faz a mesma coisa...
d1 = {10 : ‘Dez’, 20 : ‘Vinte’} # Criação já com itens...
d1[30] = ‘Trinta’ # Insere novo item com chave 30...
d1[10] = ‘Dez.’ # Se a chave já existir, muda o valor...
valor = d1[10] # Recupera o conteúdo pela chave...
valor = d1[50] # Causa erro fatal!
existe = 20 in d1 # Retorna True se chave existir.
```

Comando *for*

- O comando for pode ser usado para percorrer as chaves (e consequentemente os elementos) de um dicionário;

```
# Sintaxe do comando for para dicionários...
for ch in qualquerDicionario :
    comandoUsandoChave

dicion = {1:10, 2:20, 3:30, 4:40}
for ch in dicion : # ch recebe cada chave de dicion
    print ch, dicion [ch] # mas ordem não é garantida.
```

Tabelas sem usar dicionários

- Outra opção para implementar tabelas seria usar as representações recomendadas para arrays e registros:
  - A tabela será uma lista e seus elementos serão todos de um mesmo tipo de tupla previamente escolhido para representar o registro
  - Um dos componentes desta tupla será a chave
- Esta representação é boa quando:
  - a tabela é pequena e/ou
  - não é alterada durante a execução e/ou
  - queremos manter a tabela ordenada pelas chaves.

Exemplo completo

1. Fazer um programa em Python para:
    a. Ler uma tabela com N profissões, onde
        i. O valor de N é informado antes pelo usuário.
        ii. Cada profissão é formada por um código (número positivo) e um nome e uma área (ambos String).
    b. Depois o usuário fornecerá uma lista de códigos para que o programa informe o nome/área das profissões.
    c. Se o código da profissão não existir na tabela, mostrar a mensagem “Profissão ... não existe na tabela.” e continuar.
    d. O programa deve parar com a digitação de um código inválido (negativo ou zero).

```
# Profissões - V1 - Tabela = Dicionário com chaves e resto dos r
# Usando dicionários
n = input ('Digite o tamanho da tabela de profissões: ')
while (not isinstance (n, int)) or (n < 1) :
    n = input ('Tamanho deve ser inteiro e positivo. Tente novar
tab = { } # Criação do dicionário...
for i in range (n) :
    codP = input ('Digite o código de uma profissão: ')
    while (not isinstance (codP, int)) or (codP < 1) :
        codP = input ('Código deve ser inteiro e positivo. Tente
    nomeP = raw_input ('Digite o nome da profissão %d:\n' % (cod
    areaP = raw_input ('Digite a área da profissão %d:\n' % (cod
    tab [codP] = (nomeP, areaP) # Inserção no dicionário...
print 'Tabela com %d profissões foi lida corretamente.' % (n)
print 'Tabela ->', tab
codP = input ('Digite um código de profissão para busca (<=0 par
while codP > 0 :
    if codP in tab : # Verifica se a profissão existe na tabela
        nomeP, areaP = tab[codP] # Recupera os outros dados...
```

```
        print 'Profissão %d é %s e sua área é %s.' % (codP, nome
    else:
        print 'Profissão %d não existe na tabela.' % (codP)
    codP = input ('Digite outro código para busca (<=0 para para
print 'Fim de Programa
```

```
# -*- coding: ISO-8859-1 -*-
# Profissões – V2 - Tabela = Lista de registros com chave inclus
# Usando listas
n = input ('Digite o tamanho da tabela de profissões: ')
while (not isinstance (n, int)) or (n < 1) :
n = input ('Tamanho deve ser inteiro e positivo. Tente novamente
tab = [None]*n # Criação da lista com tamanho correto (mais efic
for i in range (n) :
codP = input ('Digite o código de uma profissão: ')
while (not isinstance (codP, int)) or (codP < 1) :
codP = input ('Código deve ser inteiro e positivo. Tente novamer
nomeP = raw_input ('Digite o nome da profissão %d:\n' % (codP))
areaP = raw_input ('Digite a área da profissão %d:\n' % (codP))
tab[i] = (codP, nomeP, areaP) # Inserção na lista...
# Omiti a impressão da tabela para caber no slide...
codP = input ('Digite um código de profissão para busca (<=0 par
while codP > 0 :
i = 0 # É preciso percorrer a lista como é feito em outras lingu
while (i < n) and (codP != tab [i] [0]) :
i = i + 1
if i < n : # Verifica se a profissão existe na tabela...
nomeP, areaP = tab [i] [1:] # Recupera os outros dados...
print 'Profissão %d é %s e sua área é %s.' % (codP, nomeP, areaP
else:
print 'Profissão %d não existe na tabela.' % (codP)
codP = input ('Digite outro código para busca (<=0 para parar):
print 'Fim de Programa'
```

Exercícios

1. Fazer um programa para:
    a. Ler uma tabela com Cursos, onde:
        i. Cada curso é formado por um código (número positivo), um nome (String), e o centro a que pertence (número positivo).
        ii. A leitura da tabela de cursos para com o código de curso zero.
    b. Depois o usuário fornecerá uma lista de códigos de centro para que o programa imprima o código e nome de todos os cursos associados a cada centro digitado.
    c. Se o código do centro não existir na tabela, mostrar a mensagem “Nenhum curso encontrado” e continuar.
    d. O programa pára com a digitação de um código de centro inválido (negativo ou zero).

```
print("1. Fazer um programa para:"
      "\nLer uma tabela com Cursos, onde:"
      "\nCada curso é formado por um código (número positiv
o), um nome (String), e o centro a que pertence (número posit
ivo)."
      "\nA leitura da tabela de cursos para com o código de c
urso zero."
      "\nDepois o usuário fornecerá uma lista de códigos de c
entro para que o programa imprima o código e nome"
      "de todos os cursos associados a cada centro digitado."
      "\nSe o código do centro não existir na tabela, mostrar
a mensagem “Nenhum curso encontrado” e continuar."
      "\nO programa pára com a digitação de um código de cent
ro inválido (negativo ou zero).\n")
areas = {100:"Exatas", 200:"Biológicas", 300:"Artes", 400:"In
formática", 500:"Engenharia"}
cursos = {}
cursoCod = int(input('Digite o código do curso: '))
```

```
while (not isinstance(cursoCod, int)) or (cursoCod <= 0) :
    cursoCod = int(input('O primeiro código não pode ser meno
r que 1. Digite o código do curso: '))
while cursoCod != 0:
    cursoNome = input(f'Digite o nome do curso {cursoCod}:')
    print("Áreas disponíveis:", areas)
    cursoArea = int(input(f'Digite a área do curso {cursoCo
d}:'))
    while not(cursoArea in areas):  # Verifica se a área exis
te...
        cursoArea = int(input(f'Área inexistente. Digite uma
área válida para o {cursoCod}:'))
    cursos[cursoCod] = (cursoNome, cursoArea)  # Inserção no
dicionário...

    # Começa a ler outro curso:
    print("\nDigite 0 para parar a leitura.")
    cursoCod = int(input('Digite o código do curso: '))
    while cursoCod in cursos or cursoCod < 0:
        cursoCod = int(input('Código inválido. Digite o códig
o do curso: '))
print('Visualizar cursos: ', cursos, "\n")

areaCod = int(input('Digite um código de área para mostrar se
us cursos (<=0 para parar): '))
while areaCod > 0:
    if areaCod in areas:
        print(f"\nA área {areaCod} possui os seguintes curso
s:")
        for chave in cursos:
            cursoNome, cursoArea = cursos[chave]
            if cursoArea == areaCod:
                print(f"- {chave}, {cursoNome}")
    else:
        print("Nenhum curso encontrado.")
```

```
    areaCod = int(input('\nDigite um código de área para most
rar seus cursos (<=0 para parar): '))
```