Paula Cardoso

Algoritmos

Registros - Dicionários (dict)

Contextualizando...

- Imagine que estamos programando um sistema de vendas de produtos agrícolas e precisamos guardar os nomes dos produtos e o preço de cada um.

produto	preço
alface	R$ 3.45
batata	R$ 1.20
tomate	R$ 2.30
feijão	R$ 1.50

- Dada uma tabela contendo preços de mercadorias, como podemos representá-la?

Contextualizando...

- Uma solução seria utilizar 2 listas: uma de strings para os produtos e outra de float para os preços

- Teríamos que associar que o produto[0] tem seu preço na lista preco[0]

produto	preço
alface	R$ 3.45
batata	R$ 1.20
tomate	R$ 2.30
feijão	R$ 1.50

Contextualizando...

- Utilizando 2 listas...

produto	preço
alface	R$ 3.45
batata	R$ 1.20
tomate	R$ 2.30
feijão	R$ 1.50

```
1. produto = ["alface", "batata", "tomate", "feijão"]
2. preco = [3.45, 1.20, 2.30, 1.50]
3. print ("O %s custa R$ %.2f" %(produto[0], preco[0]))
```

```
#ou
print ("O ",produto[0], "custa R$ ", preco[0]))
```

Contexto

- Como **dados de diferentes tipos**, porém **relacionados semanticamente** a uma mesma entidade do mundo real, devem ser armazenados em um programa de computador?
- Por exemplo: cada **pessoa** possui um nome, um número de CPF, características físicas como cor dos olhos, altura, idade, entre outras informações que a definem de maneira única.
- A questão é: como **armazenar de forma adequada informações variadas de tipos diferentes** que se referem a uma única entidade (como estas que descrevem uma pessoa, por exemplo)?
- Poderíamos usar variáveis diferentes para cada informação. Mas, o que fazer caso seja necessário armazenar dados de várias pessoas?

Contexto

Usando várias listas >>> **desvantagem**: um erro na indexação de apenas uma das listas seria suficiente para tornar todo o conjunto de dados inconsistente.

Solução desejada: seria se uma única posição de uma dada lista fosse capaz de armazenar simultaneamente todas as informações referentes a um indivíduo

Uma das formas de construir uma solução desse tipo é por meio de **registros.**

Registros

Para exemplificar, imagine aquele *ticket* de passagem que um passageiro entrega para o motorista quando vai viajar para outra cidade.
Ele é formado por informações de tipos diferentes, porém com uma conexão lógica.
Uma variável do tipo registro é uma variável composta, pois engloba um conjunto de dados, e heterogênea, pois os dados são de tipos diferentes: inteiros, float, strings, listas etc

Número da passagem: _____ Data: ____/____/_____
De: _____ Para: _____
Horário: _____ Poltrona: _____ Idade: _____
Nome do passageiro: _____

Todos os passageiros devem preencher tais dados.

Exemplo - pseudocódigo

```
Tipo
   passageiro = registro
        nome : literal / caractere
        origem : literal / caractere
        destino : literal / caractere
        bilhete: inteiro
…
Var
        p: passageiro

Inicio
        p.nome="Ada Lovelace"
        p.origem=”Marabá”
        p.destino=”Altamira”
        p.bilhete=1234

FimAlgoritmo
```

- Temos a possibilidade de criar um tipo de dado a partir de tipos existentes.
- No exemplo, criamos o tipo de dado chamado **passageiro** que é do tipo registro.
- A variável **p** é um registro do tipo pessoa.
- p tem os **campos** nome, origem, destino, bilhete

Como registros são chamados em uma linguagem?

Pascal	C, C++	Java, C++, Python	Python
record	struct	class	Dicionários (dict) Dataclass

Dicionários

- São também chamados de **vetores associativos**.
- Consistem em uma estrutura de dados similar às listas, mas com propriedades de acesso diferentes.
- Cada elemento de um dicionário é composto por um par ***chave : valor***
    - O dicionário em si consiste em relacionar uma chave a um valor específico.
- **Um dicionário não pode ter duas chaves iguais**.

Criando um dicionário

- Criamos um dicionário usando chaves { }.
- Cada **par** (*chave : valor*) deve ser separado por vírgula.

```
{"nomeDaChave1":valor , "nomeDaChave2":valor}
```

  - Após o nome da chave, aparece o operador dois-pontos seguido do valor
  - O **nome da chave** aparece entre aspas duplas.
  - O valor associado a uma chave pode ser qualquer dos tipos conhecidos (int, float, string, listas)

Criando um dicionário

- Voltando à tabela de preços de mercadorias, ela pode ser vista como um dicionário

produto	preço
alface	R$3.45
batata	R$ 1.20
tomate	R$ 2.30
feijão	R$ 1.50

- **Representação em python:**

```
tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
```

Criando um dicionário

- Representação em python:

```
tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
```

Nome da variável do tipo dicionário

Chave: valor

Criando um dicionário

Podemos ter produtos com o mesmo preço, mas não temos produtos repetidos, logo, o nome do produto será a chave!

produto	preço
alface	R$ 3.45
batata	R$ 1.20
tomate	R$ 2.30
feijão	R$ 1.50

- **Representação em python:**

```
tabela = {"alface": 0.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
```

Criando um dicionário

- Para criar um dicionário vazio, utiliza-se como delimitador os símbolos de chaves da seguinte forma:

```
agenda = {}
```

- Para inserir um valor, deve-se informar o nome da chave entre símbolos de colchetes e aspas na qual o valor será inserido:

```
agenda["Bia"] = 1234
```

Acesso aos elementos de um dicionário

- Diferente de listas (acesso por índice), um dicionário é acessado por suas **chaves**.
- Para acessar uma chave específica, seu valor deve ser informado entre colchetes.

```
1. lista =[10, 30, 8, 29]
2. print(lista[1])
3. tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
4. print (tabela["batata"])
```

Acessando um elemento de uma lista

Acessando um elemento de um dicionário

Imprimindo um dicionário (1)

```
1.  tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
2.  print(tabela)
```

```
{'alface': 0.45, 'batata': 1.2, 'tomate': 2.3, 'feijao': 1.5}
```

Imprimindo um dicionário (2)

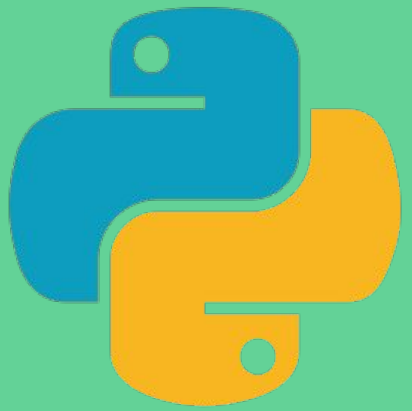
- Podemos utilizar a estrutura for para iterar pelo dicionário.
- Neste exemplo, estamos imprimindo o valor de cada chave.

```
1.  tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
2.  for i in tabela:
3.      print(tabela[i])
```

Não precisamos saber os nomes das chaves ou quantos elementos existem.
A cada rodada a variável i será uma chave

```
3.45
1.2
2.3
1.5
```

Cuidados com dicionários



Acesso a chave inexistentes

- O que acontece se acessamos uma chave que não existe?
- Exemplo:

```
1.  tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
2.  print (tabela["Batata"])
```

```
Traceback (most recent call last):
  File "feira.py", line 2, in <module>
    print (tabela["Batata"])
KeyError: 'Batata'
```

Batata e ***batata*** são tratadas como chaves diferentes!

Acesso a chave inexistentes

- O que acontece se acessamos uma chave que não existe?
- Exemplo:

```
1.  tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
2.  print (tabela[0])
```

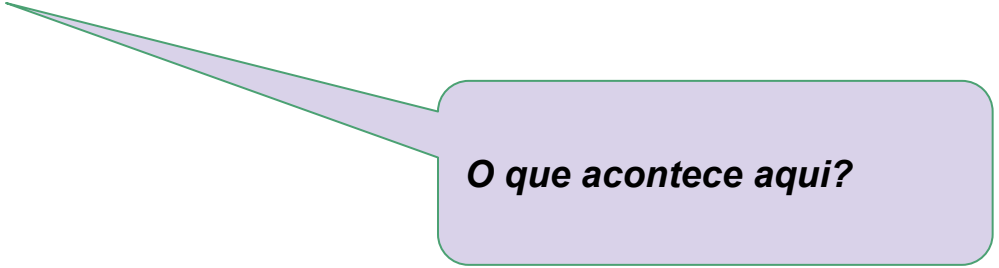
```
Traceback (most recent call last):
  File "feira.py", line 6, in <module>
    print (tabela[0])
KeyError: 0
```

Não existe posições na estrutura dicionário!

E?

- O que acontece se acessamos uma chave que não existe?
- Exemplo:

```
1. tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
2. tabela ["feijão"] = 10.0
```



Alterações em um dicionário

- Quando atribuímos um valor a uma chave, duas coisas podem ocorrer:
  - **Se a chave já existe**: o valor associado é alterado para o novo valor
  - **Se a chave não existe**: a nova chave será adicionada ao dicionário

Alterações em um dicionário

```
1.  tabela = {"alface": 3.45, "batata": 1.20, "tomate": 2.30, "feijao": 1.50}
```

- Exemplo: aumentar o valor do alface em 10% do valor atual:

```
tabela["alface"] = tabela["alface"] * 1.1
# neste caso, a chave existe, logo, estamos modificando o valor associado
```

- Exemplo: inserir na tabela o preço da cebola por R$1.30

```
tabela["cebola"] = 1.30
# neste caso, a chave não existe, logo, estamos inserindo uma
# nova entrada na tabela
```

Exemplo

Dado o dicionário telefone, cadastre um novo contato cuja chave será Mateus e o número de telefone será informado pelo usuário.

```
1. telefone={"Joao": "123-256", "Bia":"123-478"}
2. telefone["Mateus"]=input()
3. print(telefone)
```

```
paula@ubuntu-p:~/Documentos$ python3 telefone.py
345-890
{'Mateus': '345-890', 'Bia': '123-478', 'Joao': '123-256'}
```

Exemplo 1

Elabore um programa que utilize um vetor associativo (dicionário) para armazenar as informações de um dado carro.
As informações a serem armazenadas são: fabricante, modelo e ano.

Seu programa deverá então ler a informação do ano atual e informar se o carro é novo (o ano fabricação é o mesmo do ano atual), seminovo (o ano fabricação é superior aos últimos 4 anos), ou velho (o ano fabricação é anterior aos últimos 4 anos)

Código - exemplo

```
1   carro = {}
2   carro["fabricante"] = input("Qual o fabricante: ")
3   carro["modelo"] = input("Qual o modelo: ")
4   carro["ano"] = int(input("Qual o ano do carro: "))
5
6   ano_atual = int(input("Qual o ano atual: "))
7
8   if (carro["ano"] == ano_atual):
9       print("NOVO")
10  elif (carro["ano"] > (ano_atual - 4)):
11      print("SEMINOVO")
12  else:
13      print("VELHO")
```

Exemplo 2

Escreva um programa que gere um dicionário, onde cada chave seja uma vogal, e seu valor seja o número de vezes que aquela vogal é encontrada em uma frase lida.

- Exemplo de entrada: O rato
- Exemplo de saída: {“a”: 1, “e”: 0, “i”: 0, “o”: 2, “u”: 0}

Exercício

1. Eleição dos representantes anuais. Escreva um programa para facilitar a escolha dos alunos representantes do terceiro ano de uma escola. Seu programa deverá possuir um dicionário cujas **chaves** serão os nomes dos candidatos (serão **5** candidatos possíveis: J**oao, Maria, Chico, Luisa e Jose**) e os valores das chaves serão as quantidades de votos que cada candidato recebeu.
Como entrada, seu programa receberá a quantidade de votos a serem lidos e os votos. Como saída, seu programa deverá exibir os nomes dos candidatos e a quantidade de votos que eles receberam, o nome do representante (quem obteve mais votos) e o nome do vice-representante (o segundo com mais votos). Assuma que não haverão candidatos com a mesma quantidade de votos.

Paula Cardoso

Algoritmos

Lista de dicionários

Lista de dicionários

- Em Python podemos ter listas nas quais os elementos são estruturas de dados do tipo dicionário.

```
aluno = {"Nome": "Joao das Neves",
         "Matricula": 88888888,
         "Media": 30.5}

aluno1 = {"Nome": "Ana dos Paranaue",
          "Matricula": 77777777,
          "Media": 86.83}

aluno2 = {"Nome": "Bruce Wayne",
          "Matricula": 66666666,
          "Media": 100}

turma = [aluno, aluno1, aluno2]
```

O que faz a instrução **`print ( turma[1])`** ?

Lista de dicionários

- Mas e quando você não pode declarar uma variável diferente para cada dicionário?
- Pode-se fazer:

```
1. lista=[]
2. total=int(input("Quantos alunos deseja cadastrar "))
3. for i in range (total):
4.         lista.append({})      #adiciona um dicionário vazio na lista
5.         lista[i]["Nome"]=input() #adiciona a chave Nome e um valor na lista[i]
6.         lista[i]["Idade"]=int(input()) #adiciona a chave Idade e um valor na lista[i]
7. print(lista)
```

Exemplo 1

Crie um dicionário representando um atleta. Esse dicionário deve conter o nome do atleta, seu esporte, altura e idade. Leia os dados de 5 atletas e guarde em uma lista.
Imprima o nome do atleta mais velho e do atleta mais alto.

Código-exemplo

```
1   lista = []
2   for i in range(5):
3       lista.append({"nome": "", "esporte": "", "idade": 0, "altura": 0.0})
4
5   mais_alto = 0 #indice na lista do mais alto
6   mais_velho = 0 #indice na lista do mais velho
7
8   for cont in range(5):
9       print("Informacoes do atleta", cont+1)
10      lista[cont]["nome"] = input("Nome: ")
11      lista[cont]["esporte"] = input("Esporte: ")
12      lista[cont]["idade"] = int(input("Idade: "))
13      lista[cont]["altura"] = float(input("Altura: "))
14      if lista[cont]["idade"] > lista[mais_velho]["idade"]:
15          mais_velho = cont
16      if lista[cont]["altura"] > lista[mais_alto]["altura"]:
17          mais_alto = cont
18
19  print()
20  print("Mais alto:", lista[mais_alto]["nome"], "com", lista[mais_alto]["altura"])
21  print("Mais velho:", lista[mais_velho]["nome"], "com", lista[mais_velho]["idade"]
```