DICIONÁRIOS

OBS: Em algumas linguagens de programação, os dicionários são conhecidos por mapas.

Dicionários são coleções do tipo chave/valor, representados por chaves {}

print(type({}))

>>> <class 'dict'>

OBS: Sobre dicionários:

- Chave e valor são separados por 2 pontos 'chave:valor';

- Chave pode ser qualquer tipo imutavél, valor pode ser qualquer tipo de dado;

Ex: (int, float, string, bool)

- Podemos misturar tipos de dados(type de key != value)

Criando um dicionário:

Ex1:

>>> d = { } # Foi criado um dicionário vazio

>>> d

{ }

--------------------------------------------------------------------------------------------------------------------

Um dicionário com elementos:

Ex1:

>>> d = {'nome': 'Rubem', 'sobrenome': 'Álecks'}

>>> d

{'nome': 'Rubem', 'sobrenome': 'Álecks'}

CRIANDO NOVAS CHAVES E VALORES

>>> dic\_test = {‘chave0’: ‘test0’}

>>> dic\_test

{'chave0': 'test0'}

Ok....

Agora para adicionar uma new key and values:

>>> dic\_test['chave1'] = 'test1'

>>> dic\_test

{'chave0': 'test0', 'chave1': 'test1'}

ACESSANDO DADOS EM DICIONÁRIOS e METODO GET

FORMA 1

Podemos também solicitar uma informação direta ao dicionário criado, usando uma chave especifica. Exemplo:

>>> d = {'nome': 'Rubem', 'sobrenome': 'Álecks'}

>>> d['sobrenome']

'Álecks'

---------------------

# Neste caso precisamos indicar a chave através de colchetes [ ] assim como nas listas. Note que a chave fica entre aspas ‘ ’. A chave pode ser qualquer tipo de imutável, como é o exemplo das tuplas, mas não pode ser uma lista por outro lado. Já o valor pode ser qualquer objeto de valor existente no python.

Ex1:

>>> dic = {('uma\_tupla'): 'qualquer valor'}

>>> dic['uma\_tupla']:

'qualquer valor'

# note que mesmo sendo uma tupla a chave definida é usada da mesma forma, como se fosse uma string, mas caso a chave fosse um numeral (inteiro/float) não usamos as aspas ‘ ’. Caso tente usar aspas para uma chave que é um numeral será retornado um KeyError.

Ex2:

paises = {'br': 'Brasil', 'eua': 'Estados Unidos', 'py': 'Paraguai'}

print(paises['eua'])

Estados Unidos

# Caso tentemos fazer um acesso de uma chave inexistente teremos um 'KeyError'

FORMA 2:

.GET

Acessando via get - FORMA RECOMENDADA

print(paises.get('br'))

print(paises.get('ru')) # caso não encontre a chave retorna None, sem gerar KeyError

# Recomenda-se usar sempre o método get

print(paises.get('ru', 'Não Encontrei'))

~ .get faz a solicitação e caso não ache a key especifica (‘key’) e caso ele não ache retorna o segundo valor (‘Não Encontrei’)

~ caso não determine o 2º elemento por padrão get retorna None, assim evitando um KeyError.

CONSULTANDO CHAVES EM DICIONÁRIOS

in

>>> print('br' in paises)

True

>>> print('ru' in paises)

False

>>> print('Estados Unidos' in paises) # False, ele busca key e não valor. Estados Unidos é um value e não uma key.

False

...sendo assim podemos usar o if/else tendo em vista do retorno True/Else

CHAVES

Podemos utilizar qualquer tipo de dado inclusive (int, float, string, bool, tuplas)

Tuplas por exemplo são bastante interessantes como chaves, pois são imutáveis

Ex:

localidades = {

(35.6895, 39.6917): 'Escritório em Tokyo',

(40.7128, 74.0060): 'Escritório em Nova York',

(35.7749, 122.4194): 'Escritório em São Paulo'

}

ADICIONANDO ELEMENTOS

receita = {'jan': 100, 'fev': 120, 'mar': 300}

# Forma 1:

receita['abr'] = 350 # dicionario\_existente[‘Nova Chave’] = valor

print(receita)

# Forma 2:

novo\_dado = {'mai': 500}

receita.update(novo\_dado) # receita.update({'mai': 500})

ATUALIZAR DADOS EM UM DICIONÁRIO

# Forma 1

receita['mai'] = 550

# CONCLUSÃO 1: A forma1 de adicionar novos elementos ou atualizar dados é a mesma

# Forma 2

receita.update({'mai': 500})

# CONCLUSÃO 2: Em dicionários, NÃO podemos ter chaves repetidas.

REMOVENDO DADOS DE UM DICIONÁRIO

receita = {'jan': 100, 'fev': 120, 'mar': 300}

# Forma 1 - MAIS COMUM

ret = receita.pop('mar')

print(ret) # ‘mar’

print(receita)

# Precisamos sempre informar a chave, caso não encontre o elemento um KeyError é retornado

# Ao removermos um objeto desta forma(pop) o valor deste é sempre retornado(ex:ret)

# Forma 2

del receita['fev'] # NESTE CASO O VALOR REMOVIDO NÃO É RETORNADO

print(receita)

# Precisamos sempre informar a chave, caso não encontre a chave um KeyError é retornado.

EXEMPLOS PRATICO

# Imagine que você tem um comércio eletrônico, onde temos um carrinho de compras no qual adicionamos produtos.

'''Carrinho de Compras:

Produto 1:

- nome;

- quantidade;

- preço;

Produto 1:

- nome;

- quantidade;

- preço;

'''

-------------------

# 2 - Poderíamos utilizar uma Lista para isso? Sim.

carrinho = []

produto1 = ['Playstation 4', 1, 2300.00]

produto2 = ['God of War 4', 1, 150.00]

carrinho.append(produto1)

carrinho.append(produto2)

print(carrinho)

# Teríamos que saber qual era o índice de cada informação no produto

------------------------

# Forma 2 - Poderíamos utilizar uma tupla

produto1 = ('Playstation 4', 1 , 2300)

produto2 = ('God of War 4', 1, 150.00)

carrinho = (produto1, produto2)

print(carrinho)

-------------------------

# Forma 3 (CORRETA) - DICIONÁRIO

carrinho = []

produto1 = {'nome': 'Playstation 4', 'quantidade': 1, 'preço': 2300.00}

produto2 = {'nome': 'God of War 4', 'quantidade': 1, 'preço': 150.00}

carrinho.append(produto1)

carrinho.append(produto2)

print(carrinho)

# Desta forma facilmente adicionamos ou removemos produtos no carrinho

# e em cada produto podemos ter a certeza sobre cada informação

-------------------------------------------------------

# Métodos de dicionários

#>>> dir({}) - ver

>>> d = dict(a=1, b=2, c=3)

>>> print(d)

>>> print(type(d))

------------------

# Limpar o dicionário (Zerar dados) . clear()

d.clear()

print(d) # dicionário vazio – { }

----------------------------

# Copiando um dicionário para outro

d = dict(a=1, b=2, c=3)

# Forma 1 - Deep Copy – Copia sem vinculo

novo = d.copy()

print(novo)

novo['d'] = 4

print(d)

print(novo)

-----------------------------

# Forma 2 - Shallow Copy

novo = d

print(novo)

novo['d'] = 4

print(d)

print(novo)

# Tanto 'novo' quanto 'd' são alterados

-----------------------------------------------------------

# Forma não usual de criação de dicionários (bom para criar diversas chaves com único valor)

>>> outro = {}.fromkeys('a', 'b') #{}.fromkeys('chave', 'valor')

print(outro)

print(type(outro))

usuario = {}.fromkeys(['nome', 'pontos', 'email', 'profile'],'desconhecido')

print(usuario)

print(type(usuario)) # 4 key, e 1 único dado para todos

# O método fromkeys recebe 2 parâmetros: um iterável e um valor

# Ele vai gerar para cada valor do iterável uma chave e irá atribuir a essa chave o valor informado.

veja = {}.fromkeys('teste', 'valor')

print(veja)

>>> {'t': 'valor', 'e': 'valor', 's': 'valor'}''' # ele não continuou (tes'te')

# pois em dicionários não há repetição de chave

veja = {}.fromkeys(range(1,11), 'novo')

print(veja)

ITERANDO SOBRE CHAVES E VALORES