Aula 03

TUPLAS

Tuplas

- Uma ***tupla*** é uma sequência ordenada de zero ou mais referências de objetos.
- Os itens da tupla não podem ser alterados, assim, são ***imutáveis***.
- A indexação pode ser feita com índices positivos ou negativos:

t[-5]	t[-4]	t[-3]	t[-2]	t[-1]
'venus'	-28	'green'	'21'	19.74
t[0]	t[1]	t[2]	t[3]	t[4]

Exemplo

```
4    cores = "red", "green", "blue", "yellow", "black"
5
6    print(cores[:])        #todas as cores
7    print(cores[2:])       #a partir do índice 2 (inclusive)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
('red', 'green', 'blue', 'yellow', 'black')
('blue', 'yellow', 'black')
```

Exemplo

```
 9    #definição de matriz usando Tuplas
10    mat = ((1,2,3), (4,5,6))
11    print(mat[0][0])          #elemento da linha 0 e coluna 0
12    print(mat[1])             #todos os elementos da linha 1
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
(4, 5, 6)
```

Tuplas

- ***Tuplas Nomeadas***
  - Permite referenciar os itens da tupla por um nome.
  - Similar ao uso de ***struct*** da linguagem c.

```
Pessoa = collections.namedtuple("Pessoa","id nome idade email")
```

Nome da Tupla ***Campos***

Exemplo

```
1   import os
2   import collections
3   os.system('cls')
4
5   #Tuplas Nomeadas
6   Pessoa = collections.namedtuple("Pessoa","id nome idade email")
7
8   pessoas = []
9   pessoas.append(Pessoa(1,"João da Silva", 22, "joao@joao.com"))
10  pessoas.append(Pessoa(2,"Ana Maria", 19, "ana@ana.com.br"))
11
12  for p in pessoas:
13      print("Id...: %d" %p.id)
14      print("Nome.: %s" %p.nome)
15      print("Idade: %d" %p.idade)
16      print("Email: %s \n" %p.email)
```

ATIVIDADE PRÁTICA

Atividade Prática

- ***Exercício 1***
  - ***Faça um programa que carregue um vetor de seis elementos numéricos inteiros, calcule e mostre:***
    - A quantidade de números pares
    - Quais números são ímpares
    - A soma dos números
    - O maior número
    - O menor número
    - A quantidade de números positivos

Atividade Prática

- ***Exercício 2***
  - Uma imagem é formada por pixels. Considere uma imagem com dimensão de 10 x 10 e faça uma aplicação que contenha um estrutura bidimensional com essas dimensões.
  - A seguir, para cada posição da estrutura bidimensional armazene um valor aleatório entre 0 e 255 (esses valores correspondem às tonalidades aplicadas sobre a imagem).
  - Apresente em tela os valores gerados.

Atividade Prática

- ***Exercício 3***

14. Faça um programa que receba o nome e duas notas de seis alunos e mostre o relatório abaixo:

Relatório de notas:

ALUNO	1ª PROVA	2ª PROVA	MÉDIA	SITUAÇÃO
Carlos	8,0	9,0	8,5	Aprovado
Pedro	4,0	5,0	4,5	Reprovado

- média da classe = ?
- quantidade de aprovados = ?%
- quantidade de alunos de exames = ?%
- quantidade de reprovados = ?%

Atividade Prática

- ***Exercício 4***

16. *Faça um programa que receba o nome de cinco produtos e seus respectivos* preços, calcule e mostre:

- a quantidade de produtos com preço inferior a R$ 50,00;
- o nome dos produtos com preço entre R$ 50,00 e R$ 100,00;
- a média dos preços dos produtos com preço superior a R$ 100,00.

LISTAS

Listas

- As ***listas*** são coleções de itens ***heterogêneos*** que podem ser acessados:
  - ***Sequencialmente***
    - Acesso iterativo
  - ***Diretamente***
    - Acesso Indexado

O conteúdo de uma lista é ***mutável***, ou seja, pode ser ***alterado***.

- Para definição de listas é utilizado o símbolo de colchetes:

```
lista1 = []
lista2 = [1,2,3,4,5]
lista3 = [1,2,'r','palavra', 3]
```

Exemplo

```
4   lista = [1,"a",3,"palavra",5,["x","y","z"]]
5
6   for item in lista:
7       print(item)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
a
3
palavra
5
['x', 'y', 'z']
```

Listas

- Para adicionar um novo item na lista é usado o método ***lista.append(valor)***:

```
4   lista = [1,2,3]
5   lista.append(4)
6   lista.append(5)
7
8   for item in lista:
9       print(item)
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
2
3
4
5
```

Listas

- Para incluir todos os itens de uma lista em outra lista é usado o método ***lista.extend(lista)***:

```
4   lista1 = [1,2,3]
5   lista2 = [4,5,6]
6   lista3 = []
7
8   lista3.extend(lista1)
9   lista3.extend(lista2)
10
11  for item in lista3:
12      print(item)
13
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
1
2
3
4
5
6
```

Listas

- O número de itens da lista é retornado pela função ***len(nome_da_lista).***

- Para ordenar os itens da listas é usado o método ***lista.sort()***

- Para inverter a ordem os elemento da lista é utilizado o método ***lista.reverse()***

Exemplo

```
4  def exibir(lista=None):
5      for item in lista:
6          print(item)
7
8      print("Tamanho = %d \n" %len(lista))
9
10 lista = [5,2,4,3]
11 lista.sort()
12 lista.insert(0,1);  #insere o valor 1 na posição 0
13 lista.reverse()     #inverte os elementos da lista
14 lista.remove(2)     #remove o elemento 2 da lista
15 del lista[1]        #remove a posição 1 da lista
16 exibir(lista)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
5
3
1
Tamanho = 3
```

Exemplo

```
4   letras  = ["A","B","C","D"]
5   numeros = [1,2,3]
6
7   #produto cartesiano
8   prod_cart = [(let,num) for let in letras for num in numeros]
9
10  for item in prod_cart:
11      print(item)
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
('A', 1)
('A', 2)
('A', 3)
('B', 1)
('B', 2)
('B', 3)
('C', 1)
('C', 2)
('C', 3)
('D', 1)
('D', 2)
('D', 3)
```

Listas

- Métodos para manipulação de listas

Syntax	Description
`L.append(x)`	Appends item `x` to the end of `list L`
`L.count(x)`	Returns the number of times item `x` occurs in `list L`
`L.extend(m)` `L += m`	Appends all of iterable `m`'s items to the end of `list L`; the operator `+=` does the same thing

`L.index(x,`
`start,`
`end)`
	Returns the index position of the leftmost occurrence of item `x` in `list L` (or in the *start*:*end* slice of `L`); otherwise, raises a `ValueError` exception

`L.insert(i, x)`	Inserts item `x` into `list L` at index position `int i`
`L.pop()`	Returns and removes the rightmost item of `list L`
`L.pop(i)`	Returns and removes the item at index position `int i` in `L`
`L.remove(x)`	Removes the leftmost occurrence of item `x` from `list L`, or raises a `ValueError` exception if `x` is not found
`L.reverse()`	Reverses `list L` in-place
`L.sort(...)`	Sorts `list L` in-place; this method accepts the same *key* and *reverse* optional arguments as the built-in `sorted()`

Listas

- ***List comprehensions***
  - Listas de compreensões ou Abrangências de Listas
  - Permite produzir uma lista a partir de qualquer objeto iterável
  - Economiza o uso de estrutura de repetições explícitas

Exemplo

```
4   # lista de anos bissexto entre 2000 e 2030
5   anos = []
6
7   #lista tradicional
8   for a in range(2000,2030):
9       if(a % 4==0 and a % 100 !=0) or (a%400==0):
10          anos.append(a)
11
12  #list comprehension
13  anos = [a for a in range(2000,2030)
14          if(a % 4==0 and a % 100 !=0) or (a%400==0)]
15
16  print(anos)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
[2000, 2004, 2008, 2012, 2016, 2020, 2024, 2028]
```

ATIVIDADE PRÁTICA

Atividade Prática

- ***Exercício 1***
  - Create a function ***checkPalindrome*** that given the string, check if it is a palindrome.
  - Example
    - For inputString = "aabaa", the output should be checkPalindrome(inputString) = true;
    - For inputString = "abac", the output should be checkPalindrome(inputString) = false;
    - For inputString = "a", the output should be checkPalindrome(inputString) = true.

Adapted from <codefighters.com>

Atividade Prática

- ***Exercício 2***

25. Faça um programa que leia um vetor de 15 posições de números inteiros e divida todos os seus elementos pelo maior valor do vetor. Mostre o vetor após os cálculos.

Atividade Prática

- ***Exercício 3***
  - Given an array of integers, find the pair of adjacent elements that has the largest product and return that product.
  - Example
    - For inputArray = [3, 6, -2, -5, 7, 3], the output should be ***adjacentElementsProduct(inputArray) = 21***.
  - 7 and 3 produce the largest product.

Adapted from <codefighters.com>

DICIONÁRIOS

Dicionários

- Os dicionários são coleções desordenadas de pares ***<chave,valor>***
- A estrutura de dicionário é mutável, ou seja, permite alterações.
  - Entretanto, uma vez que os itens estão desordenados, não é possível determinar as posições.
- Propriedades
  - As chaves são sempre únicas e imutáveis
  - Qualquer objeto pode ser um valor

Dicionários

- Definição de um dicionário

```
dicionario1 = {}
dicionario2 = dict()
```

- Acessar um item do dicionário

```
dicionario1[chave]
```

- Atribuir ou substituir um valor

```
dicionario1[chave] = valor
```

- Remover um item

```
del dicionario1[chave]
```

Exemplo

```
6   dic = dict( {"nome":"João da Silva", "idade" : 22})
7
8   for chave,valor in dic.items():
9       print("%s = %s" %(chave,valor))
10
```

PROBLEMS OUTPUT TERMINAL ••• 1: Python Debu

```
nome = João da Silva
idade = 22
```

Syntax	Description
`d.clear()`	Removes all items from `dict d`
`d.copy()`	Returns a shallow copy of `dict d`
`d.fromkeys(s, v)`	Returns a `dict` whose keys are the items in sequence `s` and whose values are `None` or *v* if *v* is given
`d.get(k)`	Returns key `k`'s associated value, or `None` if `k` isn't in `dict d`
`d.get(k, v)`	Returns key `k`'s associated value, or `v` if `k` isn't in `dict d`
`d.items()`	Returns a view* of all the (key, value) pairs in `dict d`
`d.keys()`	Returns a view* of all the keys in `dict d`
`d.pop(k)`	Returns key `k`'s associated value and removes the item whose key is `k`, or raises a `KeyError` exception if `k` isn't in `d`
`d.pop(k, v)`	Returns key `k`'s associated value and removes the item whose key is `k`, or returns `v` if `k` isn't in `dict d`
`d.popitem()`	Returns and removes an arbitrary (key, value) pair from `dict d`, or raises a `KeyError` exception if `d` is empty
`d.setdefault(k, v)`	The same as the `dict.get()` method, except that if the key is not in `dict d`, a new item is inserted with the key `k`, and with a value of `None` or of *v* if *v* is given
`d.update(a)`	Adds every (key, value) pair from `a` that isn't in `dict d` to `d`, and for every key that is in both `d` and `a`, replaces the corresponding value in `d` with the one in `a`—`a` can be a dictionary, an iterable of (key, value) pairs, or keyword arguments
`d.values()`	Returns a view* of all the values in `dict d`

CONJUNTOS

Conjuntos

- A linguagem ***Python*** possui duas classes para manipulação de conjuntos:
  - ***set:***
    - Coleção desordenada de objetos mutáveis
  - ***frozenset:***
    - Coleção desordenada de objetos que, uma vez, criado, não pode ser alterado.

Exemplo

```
4    s1 = set("banana")
5    s2 = set("pera")
6
7    s3 = s1 | s2        #União
8    s4 = s1 & s2        #Intersecção
9    s5 = s1 - s2        #Diferença
10
11   print("União        = %s" %s3)
12   print("Intersecção = %s" %s4)
13   print("Diferença    = %s" %s5)
```

```
... 1: Python Debu ▾
União        = {'b', 'a', 'r', 'p', 'n', 'e'}
Intersecção = {'a'}
Diferença    = {'b', 'n'}
```

Syntax	Description
`s.add(x)`	Adds item `x` to set `s` if it is not already in `s`
`s.clear()`	Removes all the items from set `s`
`s.copy()`	Returns a shallow copy of set `s`*
`s.difference(t)` `s - t`	Returns a new set that has every item that is in set `s` that is not in set `t`*
`s.difference_update(t)` `s -= t`	Removes every item that is in set `t` from set `s`
`s.discard(x)`	Removes item `x` from set `s` if it is in `s`; see also `set.remove()`
`s.intersection(t)` `s & t`	Returns a new set that has each item that is in both set `s` and set `t`*
`s.intersection_update(t)` `s &= t`	Makes set `s` contain the intersection of itself and set `t`
`s.isdisjoint(t)`	Returns `True` if sets `s` and `t` have no items in common*
`s.issubset(t)` `s <= t`	Returns `True` if set `s` is equal to or a subset of set `t`; use `s < t` to test whether `s` is a proper subset of `t`*
`s.issuperset(t)` `s >= t`	Returns `True` if set `s` is equal to or a superset of set `t`; use `s > t` to test whether `s` is a proper superset of `t`*
`s.pop()`	Returns and removes a random item from set `s`, or raises a `KeyError` exception if `s` is empty
`s.remove(x)`	Removes item `x` from set `s`, or raises a `KeyError` exception if `x` is not in `s`; see also `set.discard()`

`s.symmetric_`
`difference(t)`
`s ^ t`
	Returns a new set that has every item that is in set `s` and every item that is in set `t`, but excluding items that are in both sets*

`s.symmetric_`
`difference_update(t)`
`s ^= t`
	Makes set `s` contain the symmetric difference of itself and set `t`

`s.union(t)` `s | t`	Returns a new set that has all the items in set `s` and all the items in set `t` that are not in set `s`*
`s.update(t)` `s |= t`	Adds every item in set `t` that is not in set `s`, to set `s`

ATIVIDADE PRÁTICA

Atividade Prática

- ***Exercício 1***
  - Faça um programa capaz de gerar ***usernames*** e ***senhas*** para alunos da FATEC de Ribeirão Preto.
  - O programa recebe como entrada o nome completo do aluno e produzir um *username* contendo:
    - A primeira letra do nome e o sobrenome
  - O resultado deve ser armazenado em um estrutura da sua preferência: Tupla, Lista, Dicionário ou Conjunto.
  - O programa deve garantir que não sejam gerados. *username* duplicados
  - As senhas provisórias deve conter no mínimo 8 caracteres (números, letras e símbolos) com máxima segurança.

Atividade Prática

- ***Exercício 2***
  - Modifique o exercício anterior permitindo a exibição dos dados na tela ordenado por nome:

Nome	Username	Password
Ana Maria	amaria	M%sqeR@435
Ana Clara Maria	amaria1	L(iw0#@$qerg
João da Silva	jsilva	X5M@a9owq
Marcelo Antônio	mantonio	U8q3Nrw$a@