Desenvolvimento de Sistemas Web

Prof. Fabrício Herpich

Desenvolvimento de Sistemas Web

- Revisão

Desenvolvimento de Sistemas Web

- Exercícios da aula passada

- 1) Faça um programa que imprima na tela a versão completa do verso do Chaves: “O Cão Arrependido”. O verso deve ser repetido 44 vezes.
- 2) Faça um programa em Python para exibir a tabuada de 0 a 9.
- 3) Faça um programa em Python que receba uma lista com 10 valores inteiros e mostre para o usuário qual número é o maior e qual é o menor.
- 4) Faça um programa que receba a lista a seguir e coloque em ordem crescente. Lista = [2,12,20,0,1,3,40,7,5,10].

1) Faça um programa que imprima na tela a versão completa do verso do Chaves: "O Cão Arrependido". O verso deve ser repetido 44 vezes.

```
5   verso = 0
6
7   while(verso!=44):
8       print("""
9           Volta o cão arrependido
10          Com suas orelhas tão fartas
11          Com seu osso roído
12          E com o rabo entre as patas
13      """)
14      #print(verso)
15      verso += 1
```

2) Faça um programa em Python para exibir a tabuada de 0 a 9.

```
"""
2) Faça um programa em Python para exibir a tabuada de 0 a 9

"""

for i in range(0,10):
    print("Tabuada do", i)
    for j in range(1,11):
        print(i, "*", j, "=",i*j)

```

3) Faça um programa em Python que receba uma lista com 10 valores inteiros e mostre para o usuário qual número é o maior e qual é o menor.

```
5   lista = []
6
7   for i in range(0,10):
8       lista.append(int(input("Digite um valor inteiro:")))
9
10  maior = lista[1]
11  menor = lista[1]
12
13  for item in lista:
14      #print(item)
15      if item > maior:
16          maior = item
17      if item < menor:
18          menor = item
19
20  print("Maior: ", maior)
21  print("Menor: ", menor)
```

4) Faça um programa que receba a lista a seguir e coloque em ordem crescente. Lista = [2,12,20,0,1,3,40,7,5,10].

```
"""
4) Faça um programa que receba a lista a seguir e coloque em ordem crescente.
Lista = [2,12,20,0,1,3,40,7,5,10]

"""
lista = [2,12,20,0,1,3,40,7,5,10]
lista.sort()
print(lista)
```

Desenvolvimento de Sistemas Web

- Tópicos da aula

- Introdução a Linguagem de Programação Python
  - Dicionários
  - Conjuntos
  - Funções
  - Parâmetros
  - Expressão lambda
  - Algumas funções: Map, Filter e Reduce

Desenvolvimento de Sistemas Web

❑ Dicionários

```
"""
Lista e tuplas: indexadas
lista = ["fabricio","outro"]
lista : 0 -> "fabricio", 1-> "outro"

com dicionários, é possível definir a chave.
Por isso, é comum encontrar a definição de
dicionário como sendo chave-valor.
"""
# dicionário: cria-se uma chave e atribui-se um valor
dicionario = {
    'correr': 'Deslocar-se ou mover-se rapidamente.',
    'ligar': 'Estabelecer uma comunicação.',
}
print(dicionario)
print(dicionario['correr'])
print(dicionario['ligar'])
```

Faça um teste!

Desenvolvimento de Sistemas Web

Dicionários

```
19  carro = {
20      'modelo': 'Fusca',
21      'marca': 'Volkswagem',
22      'ano': 1970,
23      'proprietarios': ['fabricio','joao','pedro']
24  }
25
26  print(type(carro))
27  print(dir(carro))
28  print(carro['modelo'])
29  print(carro['proprietarios'])
30  print(carro['proprietarios'].append("maria"))
31  print(carro)
32
33  #adicionar nova chave-valor
34  carro['Rodagem'] = 80500
35  print(carro)
```

Faça um teste!

Desenvolvimento de Sistemas Web

❑ Dicionários

```
37  #alterar valor com acesso direto
38  carro['ano'] = 1985
39  print(carro)
40
41  #alterar valor com método
42  carro.update({'ano': 1980})
43  print(carro)
44
45  #removendo item do dicionario
46  del carro['ano']
47  print(carro)
48
49  #Acessando as chaves e valores
50  print(carro.keys())
51  print(carro.values())
52  print(carro.items()) #itens separados por lista e tuplas para cada
53
54  #Recuperando valores
55  print(carro.get('modelo'))
56  print(carro.get('ano', 'padrao')) #definindo um padrão caso não encontrar um valor
```

Faça um teste!

Desenvolvimento de Sistemas Web

❑ Dicionários

```
37  #alterar valor com acesso direto
38  carro['ano'] = 1985
39  print(carro)
40
41  #alterar valor com método
42  carro.update({'ano': 1980})
43  print(carro)
44
45  #removendo item do dicionario
46  del carro['ano']
47  print(carro)
48
49  #Acessando as chaves e valores
50  print(carro.keys())
51  print(carro.values())
52  print(carro.items()) #itens separados por lista e tuplas para cada
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Dicionários

Faça um teste!

```
54  #Recuperando valores
55  print(carro.get('modelo'))
56  print(carro.get('ano', 'padrao')) #definindo um padrão caso não encontrar um valor
57
58  print(len(carro))
59
60  #limpar todos os itens
61  carro.clear()
62  print(carro)
```

Conjuntos

```
65  """
66  Estrutura set:
67  set -> conjunto de valores não indexados
68  """
69  itens = {"Fabrício", "João", "Pedro"}
70  print(type(itens))
71  print(itens)
72  itens = {"Fabrício", "João", "Pedro", "Pedro"}
73  print(itens) #só permite um único valor igual
74  # diferente de lista, que usa indexador e permite repetição de valores
75  lista = ["Fabrício", "João", "Pedro", "Pedro"]
76  print(lista)
```

Faça um teste!

Desenvolvimento de Sistemas Web

Conjuntos

```
78  # União de conjuntos
79  carros = {"Fusca", "Gol", "Fiat 147", "Opala"}
80  carros2 = {"BMW", "Fusca", "Passat"}
81  novo = carros.union(carros2)
82  print(novo)
83
84  # Interseção - retorna elementos iguais
85  novo = carros.intersection(carros2)
86  print(novo)
87
88  # Adicionando elementos
89  carros.add("BMW")
90  print(carros)
91
92  # Removendo elementos
93  carros.remove("Fiat 147")
94  print(carros)
```

Faça um teste!

Desenvolvimento de Sistemas Web

❑ Funções

```
1   # calcular a media do Fabrício
2   totalNotas = 5+7+5
3   media = totalNotas / 3
4
5   print(f'Média do Fabrício é: {media}')
6
7   # calcular a media da Ana
8   totalNotas = 5+7+5
9   media = totalNotas / 3
10
11  print(f'Média da Ana é: {media}')
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Funções – sem retorno

```
14  def calcular_media(nome, nota1, nota2, nota3):
15      totalNotas = nota1 + nota2 + nota3
16      media = totalNotas / 3
17
18      print(f'Média da {nome} é: {media}.')
19
20
21  calcular_media("Fabrício",8,9,10) # calcular a media do Fabrício
22  calcular_media("Ana",8,5,8) # calcular a media do Ana
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Funções – com retorno

```
25  def calcular_media2(nota1, nota2, nota3):
26      totalNotas = nota1 + nota2 + nota3
27      media = totalNotas / 3
28      return media
29
30  retorno = calcular_media2(9,9,10)
31  print(f'Média do Fabrício é: {retorno}')
```

Faça um teste!

Funções – passagem de lista por parâmetro

```
35  alunos = [
36      {'nome': 'Fabricio', 'notas': [10,9,8]},
37      {'nome': 'Ana', 'notas': [10,9,10]},
38      {'nome': 'Pedro', 'notas': [10,10,10]}
39  ]
40
41  def calcular_media3(notas): # recebe a lista notas por parâmetro
42      totalNotas = 0
43      for nota in notas: # percorre as notas
44          totalNotas += nota # soma o total das notas
45
46      media = totalNotas / len(notas) # dividindo pela quantidade de notas
47      return media
48
49  for aluno in alunos:
50      nome = aluno['nome'] # buscando o nome através da chave do dicionário
51      notas = aluno['notas'] # buscando as notas através da chave do dicionário
52      media = calcular_media3(notas) # chama a função
53      print(f'Média do(a) {nome} é: {media}') # apresenta a média
```

Faça um teste!

Funções – múltiplos parâmetros

```
55  # função com num. de param. estático
56  def somar(n1,n2,n3):
57      total = n1+ n2+ n3
58      print(f'total: {total}')
59
60  somar(1,1,1)
61
62  # função com múltiplos parâmetros (packing)
63  def somar(*numeros):
64      # usando asterisco é possível definir múltiplos parâmetros
65      # a função empacto esses parâmetros usando uma tupla.
66      print(type(numeros))
67      total = 0
68      for numero in numeros:
69          total += numero
70      print(f'total: {total}')
71
72  somar(5,6,7)
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Funções – múltiplos parâmetros

```
74  # múltiplos parâmetros (unpacking)
75  def somar2(numero1, numero2):
76      soma = numero1+numero2
77      print(f'Soma: {soma}')
78
79  num = [10,8] # Consegue passar uma lista, tuplas e sets
80  # se a lista passada contiver mais que dois valores, vai acusar erro
81  somar2(*num) # Unpacking da lista num. Desempacota.
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Funções – parâmetros opcionais

Faça um teste!

```
63  # parametros opcionais
64  def calcular_media(nota1, nota2, ponto_extra=0):
65      media = (nota1+nota2)/2 + ponto_extra
66      print(f'Média: {media}')
67
68  calcular_media(9,8)
```

Desenvolvimento de Sistemas Web

- Funções – parâmetros nomeados

Faça um teste!

```
70  # parametros nomeados
71  def calcular_media(nota1, nota2, ponto_extra=0, nota_extra=0):
72      media = (nota1+nota2)/2 + ponto_extra
73      print(f'Média: {media} extra: {nota_extra}')
74
75  calcular_media(9,8,nota_extra=5)
```

Desenvolvimento de Sistemas Web

- Funções – expressões lambda (função anônima)

```
"""
    Expressões lambda -> função anonima
"""
subtrair = lambda a , b: a-b
print(subtrair(5,3))

multiplicacao = lambda a, b: print(f'Resultado: {a*b}')
multiplicacao(2, 3)
```

Faça um teste!

Desenvolvimento de Sistemas Web

❑ Funções – função map

```
"""
Map -> mapear os dados, percorrer e alterá-los
percorre uma lista e você consegue executar operações
"""
lista_numeros = [10,20,30,40,50]

nova_lista = map(lambda n: n*2, lista_numeros)
print(type(nova_lista))
print(type(lista_numeros))
l = list(nova_lista)
#print(l)
print(type(l))
for item in nova_lista:
    print(item)
```

Faça um teste!

Desenvolvimento de Sistemas Web

❑ Funções – função filter

```
"""
Filter -> filtrar dados
    Retorna Verdadeiro ou Falso
Exemplo
Numero >= 20
10 - falso
20 - true
30 - true
novalista
20
30
"""

lista = [10,20,30,40,50]
nova_lista = filter(lambda n: n>=20, lista)

print(list(nova_lista))
```

Faça um teste!

Desenvolvimento de Sistemas Web

- Funções – função filter

Faça um teste!

```
19  lista = [
20      {'produto': 'fone de ouvido', 'preco': 500},
21      {'produto': 'controle xbox', 'preco': 200},
22      {'produto': 'celular', 'preco': 1000},
23  ]
24
25  nova_lista = filter(lambda p: p['preco'] >= 500, lista)
26  print(list(nova_lista))
```

Funções – função reduce

```
"""
Reduce (reduzir) -> acumula e reduz uma lista
em um único valor.

Não vem dentro do builtins, por isso é necessária
a sua importação.
"""
from functools import reduce
lista = [10,20,30,40,50]

acumula = 0
for item in lista: # como faríamos até então
    acumula += item

print(acumula)

# Agora com a função reduce: passa a função que vai realizar a operação.
# Passa a lista e o valor que o acumulador irá iniciar.
funcao = lambda acumulador, item: acumulador + item
resultado = reduce(funcao, lista, 0)
print(resultado)
```

Faça um teste!