TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 04 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva uma função em Python chamada `indice_maior_elemento` que recebe um tuplo contendo números inteiros, e devolve o índice do maior elemento do tuplo. Por exemplo,

```
>>> indice_maior_elemento((2, 4, 23, 76, 3))
3
```

Solução:

```
def indice_maior_elemento(tuplo):
    maior = 0
    for i in range(len(tuplo)):
        if tuplo[i] > tuplo[maior]:
            maior = i
    return maior
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 04 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva em Python a função `repete_duplica` que recebe um tuplo e retorna como resultado um tuplo idêntico ao original, mas em que cada elemento está repetido com a sua repetição duplicada. Por exemplo,

```
>>> repete_duplica((1, 2, 3))
(1, 2, 2, 4, 3, 6)
```

Solução:

```
def repete_duplica(tuplo):
    newtuplo = ()
    for e in tuplo:
        newtuplo = newtuplo + (e, 2*e)
    return newtuplo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 04 (30 minutos) Turno 4ª feira 08:00-09:30
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva em Python a função `indexify` que recebe um tuplo e retorna como resultado um tuplo idêntico ao original, mas em que após cada elemento é inserido o elemento multiplicado pelo seu índice no tuplo original. Por exemplo,

```
>>> indexify((1, 2, 3))
(1, 0, 2, 2, 3, 6)
```

Solução:

```
def indexify(tuplo):
    newtuplo = ()
    for i in range(len(tuplo)):
        newtuplo = newtuplo + (tuplo[i], i*tuplo[i])
    return newtuplo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 04 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva em Python a função `insere` que recebe um tuplo e um valor, e retorna como resultado um tuplo idêntico ao original, mas em que após cada elemento é inserido o valor passado como parâmetro multiplicado pelo elemento do tuplo. Por exemplo,

```
>>> insere((1, 2, 3), -2)
(1, -2, 2, -4, 3, -6)
```

Solução:

```
def insere(tuplo,a):
    newtuplo = ()
    for e in tuplo:
        newtuplo = newtuplo + (e, e*a)
    return newtuplo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 04 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva uma função em Python com o nome `menores` que recebe um tuplo contendo números inteiros e um número inteiro e que devolve um tuplo com todos os índices dos elementos do tuplo original que são menores do que esse inteiro. Por exemplo,

```
>>> menores((3, 4, 5, 6, 2), 5)
(0,1,4)
>>> menores((3, 4, 5, 6, 2), 2)
()
```

Solução:

```
def menores(tuplo, limit):
    newtuplo = ()
    for i in range(len(tuplo)):
        if tuplo[i] < limit:
            newtuplo += (i,)
    return newtuplo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 3 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva uma função em Python com o nome `maiores` que recebe um tuplo contendo números inteiros e um número inteiro e que devolve um tuplo com todos os índices dos elementos do tuplo que são maiores do que esse inteiro. Por exemplo,

```
>>> maiores((3, 4, 5, 6, 7), 5)
(3,4)
>>> maiores((3, 4, 5, 6, 7), 8)
()
```

Solução:

```
def maiores(tuplo, limit):
    newtuplo = ()
    for i in range(len(tuplo)):
        if tuplo[i] > limit:
            newtuplo += (i,)
    return newtuplo
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 3 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 4 - Tuplos e Ciclos Contados

Sem utilizar um ciclo while, escreva uma função em Python com o nome `diferentes` que recebe um tuplo contendo números inteiros e um número inteiro e que devolve um tuplo com todos os índices dos elementos do tuplo que são diferentes a esse inteiro. Por exemplo,

```
>>> diferentes((3, 4, 5, 6, 7), 5)
(0,1,3,4)
>>> diferentes((5,5,5), 5)
()
```

Solução:

```
def diferentes(tuplo, value):
    newtuplo = ()
    for i in range(len(tuplo)):
        if tuplo[i] != value:
            newtuplo += (i,)
    return newtuplo
```