TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 2ª feira 09:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe um número inteiro positivo, `numero`, e um número inteiro positivo menor que 10, `digito`, e devolve quantos dígitos no `numero` são diferentes de `digito`. A função deve verificar a validade dos seus argumentos.
Por exemplo,

```
>>> conta_ocurrencia(234567, 3)
5
```

Solução:

```
def conta_ocurrencia(num, dig):
    if type(num) != int or type(dig) != int \
        or num <= 0 or dig > 10 or dig <= 0:
        raise ValueError('invalid arguments')
    cnt = 0
    while num != 0:
        if num % 10 == dig:
            cnt += 1
        num = num // 10
    return cnt
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe um número inteiro positivo, `numero`, e um número positivo menor que 10, `digito`, e devolve a posição da primeira ocorrência do `digito` no `numero`. Considere o dígito das unidades como a posição 1. Caso não encontre, devolve 0. A função deve verificar a validade dos seus argumentos.
Por exemplo,

```
>>> primeira_ocurrencia(102345267, 4)
5
```

Solução 1:

```
def primeira_ocurrencia(num, dig):
    if type(num) != int or type(dig) != int \
        or num <= 0 or dig > 10 or dig <= 0:
        raise ValueError('invalid arguments')
    pos, i = 0, 0
    while num != 0:
        i += 1
        d = num % 10
        num = num // 10
        if pos == 0 and d == dig:
            pos = i
    return pos
```

Solução 2:

```
def primeira_ocurrencia(num, dig):
    if type(num) != int or type(dig) != int \
        or num <= 0 or dig > 10 or dig <= 0:
        raise ValueError('invalid arguments')
    pos, i = 0, 0
    while num != 0:
        i += 1
        d = num % 10
        num = num // 10
        if d == dig:
            pos = i
            break
    return pos
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe um número inteiro positivo, `numero`, e um número positivo menor que 10, `digito`, e devolve a posição da última ocorrência do `digito` no `numero`. Considere o dígito das unidades como a posição 1. Caso não encontre, devolve 0. A função deve verificar a validade dos seus argumentos.
Por exemplo,

```
>>> ultima_ocurrencia(102345267, 2)
7
```

Solução:

```
def ultima_ocurrencia(num, dig):
    if type(num) != int or type(dig) != int \
        or num <= 0 or dig > 10 or dig <= 0:
        raise ValueError('invalid arguments')
    pos, i = 0, 0
    while num != 0:
        i += 1
        d = num % 10
        num = num // 10
        if d == dig:
            pos = i
    return pos
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 5ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva a função codifica que recebe um número inteiro positivo e que devolve uma codificação para esse número do seguinte modo: (a) cada algarismo par é substituído pelo algarismo par anterior, entendendo-se que o algarismo par anterior a 0 é o 8; (b) cada algarismo ímpar é substituído pelo algarismo ímpar seguinte, entendendo-se que o algarismo ímpar seguinte a 9 é o 1; (c) o número obtido é invertido. A função deve verificar a validade dos seus argumentos.

Por exemplo,

```
>>> codifica(2095)
7180
```

Solução:

```
def codifica(num): res = 0
      if type(num) != int or num <= 0 :
            raise ValueError('invalid arguments')
      while num != 0:
            digito = num % 10
            num = num // 10
            if digito % 2 == 0:
                  digito = (digito - 2)%10
            else:
                  digito = (digito + 2) % 10
            res = res * 10 + digito
      return res
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 6ª feira 10:00-11:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe um número inteiro positivo `n` e calcula o `n`-ésimo termo da série de Fibonacci, onde cada termo é calculado através da soma dois termos anteriores: `fib(n) = fib(n-1) + fib(n-2)`, onde `fib(0)=0` e `fib(1)=1`. Não pode utilizar uma solução recursiva. A função deve verificar a validade dos seus argumentos.

Por exemplo,

```
>>> fib(12)
144
```

Solução:

```
def fib(n):
    if type(n) != int or num <= 0 :
        raise ValueError('invalid arguments')
    a, b, f, i = 0, 1, 1, 1
    while i < n:
        f = a + b
        if a < b:
            a = f
        else :
            b = f
        i += 1
    return f
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe dois números inteiros positivos e calcula o mínimo múltiplo comum. A função deve verificar a validade dos seus argumentos.
Por exemplo,

```
>>> minimo_multiplo_comum(4, 6)
12
>>> minimo_multiplo_comum(8, 6)
24
```

Solução:

```
def minimo_multiplo_comum(a, b):

      if type(a) != int or type(b) != int \
            or a <= 0 or b <= 0:
            raise ValueError('invalid arguments')
      if b > a:
            a, b = b, a
      i = a
      while i < a * b:
            if i % a == 0 and i % b == 0:
                  break
            i += 1

      return i
```

TÉCNICO LISBOA

Fundamentos de Programação - 2020/2021 Aula Prática 03 (30 minutos) Turno 6ª feira 15:30-17:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python que recebe dois números inteiros positivos e calcula o máximo divisor comum entre eles. A função deve verificar a validade dos seus argumentos.

Por exemplo,

```
>>> maximo_divisor_comum(4, 6)
2
>>> maximo_divisor_comum(18, 6)
6
```

Solução:

```
def maximo_divisor_comum(a, b):
    if type(a) != int or type(b) != int \
        or a <= 0 or b <= 0:
        raise ValueError('invalid arguments')
    d, i = 1, 1
    if a < b :
        a, b = b, a
    while i <= b :
        if a % i == 0 and b % i == 0 :
            d = i
        i = i + 1
    return d
```