TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 4ª feira 8:00-9:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Um número inteiro, `n`, diz-se *triangular* se existir um inteiro `m` tal que `n = 1 + 2 + ... + (m - 1) + m`. Escreva uma função chamada `triangular` que recebe um número inteiro positivo `n`, e cujo valor é `True` apenas se o número for triangular. No caso de `n` ser `0` deverá devolver `False`. Por exemplo,

```
>>> triangular(6)
True
>>> triangular(8)
False
```

Solução:

```
def triangular(n):
    soma = 0
    i = 1
    while soma < n:
        soma = soma + i
        i = i + 1
    return soma == n
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 6ª feira 12:30-14:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

A função *arctg* pode ser calculada através da seguinte fórmula

$$arctg(x) = \sum_{i=0}^{n} \frac{(-1)^i x^{2i+1}}{2i+1}$$

Escreva uma função com o nome `arctg`, que recebe o número `x` para o qual se quer calcular o *arctg*, bem como o número de termos `n` da expressão a calcular, e devolve o *arctg* calculado de acordo com a fórmula anterior.

```
>>> arctg(0.5, 100)
0.46364760900080615
>>> arctg(1, 100)
0.7878733502677477
```

Solução:

```
def arctg(x,n):
    soma = 0
    while n >= 0:
        termo = (-1)**n * x**(2*n +1) / (2*n + 1)
        soma = soma + termo
        n = n - 1
    return soma
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 6ª feira 14:00-15:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Um número *primo* é um número inteiro maior do que 1 que apenas é divisível por 1 e por si próprio. Por exemplo, 5 é primo porque apenas é divisível por si próprio e por um, ao passo que 6 não é primo pois é divisível por 1, 2, 3, e 6. Os números primos têm um papel muito importante tanto em Matemática como em Informática. Um método simples, mas pouco eficiente, para determinar se um número, `n`, é primo consiste em testar se *n* é múltiplo de algum número entre `2` e $\sqrt{n}$.
Usando este processo, escreva uma função em Python chamada `primo` que recebe um número inteiro e tem o valor `True` apenas se o seu argumento for primo.

```
>>> primo(9)
False
>>> primo(11)
True
```

Solução:

```
import math
def primo(n):
    i = 2
    max = math.sqrt(n)
    while i <= max:
        if n % i == 0:
            return False
        i = i + 1
    return True
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 2ª feira 9:00-10:30
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Um número *d* é divisor de *n* se o resto da divisão de *n* por *d* for 0. Escreva uma função com o nome `num_divisores` que recebe um número inteiro positivo `n`, e tem como valor o número de divisores de `n`. No caso de `n` ser 0 deverá devolver 0. Por exemplo,

```
>>> num_divisores(20)
6
>>> num_divisores(13)
2
```

Solução:

```
def num_divisores(n):
    res = 0
    i = 1
    while i <= n:
        if n%i == 0:
            res = res + 1
        i = i + 1
    return res
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 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 calcula o valor aproximado da série para um determinado valor de x:

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} = e^x$$

O cálculo do valor aproximado da série deverá terminar quando o termo a adicionar for inferior a um certo `delta` (que é fornecido como segundo parâmetro da função). Assuma a existência da função `factorial(n)`.

Solução:

```
def serie_e(x, delta):
    soma = 0
    n = 0
    termo = 1
    while termo >= delta:
        soma = soma + termo
        n = n + 1
        termo = x**n / factorial(n)
    return soma
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 2ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

Escreva uma função em Python com o nome `soma_quadrados` que recebe um número inteiro positivo, `n`, e tem como valor a soma dos quadrados de todos os números inteiros de 1 até `n`.

```
>>> soma_quadrados(1)
1
>>> soma_quadrados(3)
14
```

Solução:

```
def soma_quadrados(n):
    soma = 0
    while n != 0:
        soma = soma + n*n
        n = n - 1
    return soma
```

TÉCNICO LISBOA

Fundamentos de Programação - 2018/2019 Aula Prática 03 (30 minutos) Turno 5ª feira 10:30-12:00
Nome:
Número:
Data:
Curso:

Capítulo 3 - Funções

O número máximo de fatias de pizza possível de se obter com apenas `n` cortes[1], é dado pela expressão `(n*n + n + 2) / 2`. Escreva uma função em Python com o nome `num_cuts` que recebe o número de fatias necessário `slices`, e devolve o número mínimo de cortes que são necessários para as produzir. Por exemplo:

```
>>> num_cuts(1)
0
>>> num_cuts(22)
6
```

Solução 1:

```
def num_cuts(slices):
    n = 0
    f = 1
    while f < slices:
        n = n + 1
        f = (n*n + n + 2) / 2
    return n
```

Solução 2:

```
def num_cuts(slices):
    n = 0
    while formula(n) < slices:
        n = n + 1
    return n

def formula(n):
    return (n*n + n + 2) / 2
```

[1] https://en.wikipedia.org/wiki/Lazy_caterer's_sequence