**Prática 2**

1. Escreva a função multiplica() que, dados dois números, execute a sua multiplicação e imprima o resultado devidamente identificado.   
   >>>multiplica(2,3)  
   >>>Resultado de multiplicar 2 com 3: 6
2. O Python tem uma função "built-in" que devolve o número de caracteres numa string, chamada len(). Exemplo: len('Hello') devolve o valor 5. Escreva uma função chamada right\_justify(), que recebe uma string s como argumento, e escreve a string com espaços suficientes à esquerda, de modo a que o último caracter da string fique na coluna 70 da consola.   
   >>>right\_justify(‘Programação I')  
   >>>’  
   Programação I’
3. Escreva uma função que recebe como argumento um número inteiro e escreve o algarismo das unidades desse número. Por exemplo:  
   >>>unidades(123)  
   >>>O numero 123 possui 3 como algarismo das unidades  
   Dica: o operador % calcula o resto da divisão dos seus operandos.

**Prática 3**

1. Implemente funções que recebem um inteiro positivo n e escrevem
   1. todos os inteiros de 0 a n-1
   2. a soma dos inteiros de 1 a n

**Prática 4**

1. Defina uma função que verifica se um número é par.  
   >>> numero\_par(2)  
   >>> True
2. Escreva uma função que recebe um número inteiro positivo e escreve True se tal número só tiver apenas um algarismo e False de outro modo.  
   >>> um\_algarismo(8)  
   >>> True  
   >>> um\_algarismo(12)  
   >>> False
3. Escreva uma função que recebe uma string e dois booleanos. A string indicará a porta lógica (AND ou OR) e posteriormente apresentará o resultado da conjugação dos dois booleanos.  
   >>>portas\_logicas('and', True, True)  
   >>>True
4. Escreva uma função que recebe três números inteiros diferentes e escreve o número do "meio" (aquele fica entre os outros dois).  
   >>> numero\_meio(20, 10, 30)  
   >>> 20  
   Dica: Utilize as funções max() e min() pré-definidas no Python  
   max(10,20,30) -> 30  
   min(10,20,30) -> 10
5. Escreva uma função que receba três valores. A função deverá informar se os valores formam um triângulo. Se sim a função deverá indicar que tipo de triângulo.  
   >>> triangulo(3,9,6.5)  
   >>> Os valores inseridos 3, 9, 6.5 formam um triângulo  
   escaleno  
   Dicas:  
   • Existe um triângulo quando a soma de quaisquer dois lados  
   for maior que o terceiro;  
   • Triângulo Equilátero: três lados iguais;  
   • Triângulo Isósceles: quaisquer dois lados iguais;  
   • Triângulo Escaleno: três lados diferentes;
6. Defina a função ano\_bissexto que verifique se um ano é bissexto.  
   Defina a função data\_valida que peça um dia, mês e ano e use a função ano\_bissexto para ver se uma data é válida.  
   >>> ano\_bissexto(1900)   
   >>> False   
   >>> ano\_bissexto(2012)   
   >>> True   
   >>> data\_valida(29,2,2000)  
   >>> Data Válida  
   >>> data\_valida(29,2,1900)  
   >>> Data Inválida
7. Factorial
   1. Defina a função factorial de forma recursiva
   2. Defina a função factorial de forma iterativa
   3. Desenhe o “stack diagram” para a forma recursiva de factorial(4).  
      Dica:  
      - factorial (0) = 1; factorial (1) = 1;  
      - factorial(n) = n \* factorial(n-1);  
      - Verificar se o numero introduzido é inteiro, utilizando a função  
      isinstance();  
      - Verificar se o número introduzido é maior que zero.  
      - factorial(4) = 4\*3\*2\*1 = 24
8. Fibonnaci
   1. Defina a função fibonacci de forma recursiva
   2. Defina a função fibonacci de forma iterativa
   3. Desenhe o “stack diagram” para a forma recursiva de fibonacci(5).  
      Dica:
   4. - fibonacci (0) = 0; fibonacci (1) = 1;  
      - fibonacci(n) = fibonacci(n-1) + fibonacci(n-2);  
      - Verificar se o numero introduzido é inteiro, utilizando a função  
      isinstance();  
      - Verificar se o número introduzido é maior que zero.  
      - Sequência de fibonacci: 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, ...

**Prática 5**

**1.** Escreva uma função denominada compara(x, y) que devolve 1 se x > y, 0 se x == y e -1 se x < y.

>>> compara(1,1)

>>> 0

**2.** Utilizando desenvolvimento incremental (ver slides de "Funções com retorno"), escreva uma função  hipotenusa(cateto1, cateto2) que retorna o comprimento da hipotenusa de um triângulo retângulo dados os comprimentos dos dois catetos.

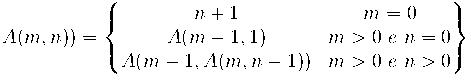
>>> hipotenusa(4,3)

>>> 5.0

**Dica:**

* [h^2 = a^2+b^2](https://www.moodle.uevora.pt/1415/filter/tex/displaytex.php?texexp=h%5E2%20%3D%20a%5E2%2Bb%5E2)
* Utilize as funções math.sqrt(n) para calcular a raiz quadrada e round(n, nº casas decimais) para arredondar o resultado.

**3.** Construa a função Ackermann de forma recursiva, A(m,n) é definida por:

[](https://www.moodle.uevora.pt/1415/filter/tex/displaytex.php?texexp=A%28m%2Cn%29%29%20%3D%20%5Cbegin%7BBmatrix%7Dn%20%2B%201%20%26%20m%20%3D%200%5C%5C%20A%28m-1%2C1%29%20%26%20m%20%3E%200%20%5C%20e%5C%20n%20%3D%200%5C%5C%20A%28m-1%2CA%28m%2Cn-1%29%29%20%26%20m%20%3E%200%20%5C%20e%5C%20n%3E0%5Cend%7BBmatrix%7D)

>>> ack(3, 4)

>>> 125

**Dica:**

•Realizar a verificação se são introduzidos números inteiros, utilizando a função isinstance() e não negativos.

**4.** Analise as seguintes funções, construa o stack diagram e diga qual é o resultado:

def b(z):

    prod = a(z, z)

    print(z, prod)

    return prod

def c(x, y, z):

    sum = x + y + z

    pw = b(sum)\*\*2

    return pw

def a(x, y):

    x = x + 1

    return x \* y

x = 1

y = x + 1

print(c(x, y+3, x+y))

**5.** Um "palíndromo" é uma palavra que é escrita da mesma forma da esquerda para a direita ou da direita para a esquerda, como por exemplo aia e noon. De um modo recursivo podemos dizer que uma palavra é um palíndromo  se a primeira e a última letra são iguais e o "meio da palvra" é um palíndromo.

    def first(word):  
        return word[0]  
  
    def last(word):  
        return word[-1]  
  
    def middle(word):  
        return word[1:-1]

Experimente cada uma das funções acima. O que ocorre se chamarmos middle com um palavra somente com 2 letras? E com 1? E sem letras?

Escreva uma função palíndromo que recebe como argumento uma string e devolve True se for um palíndromo e False de outro modo.

**Dica:**

* A função len() permite saber o tamanho de uma string.
* Lista de palíndromos: aia, ala, ama, Ana, arara, matam, radar, reger, rever, reviver, rotor…

**6.** O máximo divisor comum (MDC) entre A e B é o maior número que é divisor de A e de B ao mesmo tempo.

Uma forma de calcular o MDC é dada pelo algoritmo de Euclides, que se baseia na observação de que se R é o resto da divisão de A por B, então MDC(A,B) = MDC(B,R). Escreva uma função mdc que recebe como argumentos dois inteiros A e B e devolve o máximo divisor comum entre os dois.

>>> mdc(348, 156)

>>> 12

**Dica:**

* Como caso base, podemos considerar MDC(A,0) = A.
* Verificar se são introduzidos dois números inteiros.

**7.** Utilizando iteração, reescreva a função abaixo:

    def print\_n(s, n):

        if n <= 0:

            return

        print(s)

        print\_n(s, n-1)

**8.** Utilize a raiz quadrada dada na aula teórica, realize a comparação da mesma com a função math.sqrt(). Crie uma função test\_sqrt(a,x), que receba dois números (sendo a o número a calcular e x a estimativa), o output deverá ser de 1 até à variável a e dividido em 4 colunas: o numero a, raiz quadrada da aula, math.sqrt() e o valor absoluto da diferença entre os 2 valores anteriores.

>>> test\_sqrt(3,3)

>>>   Numero  RQ      SQR     VA

         1            1.0       1.0        0.0

         2            1.41…  1.41…  2.220…

         3            1.73…  1.73…  0.0

**Dica:**

* Exemplo para formatar linhas: print('{0:<12} {1:<20} {2:<20} {3:<20}'.format('Numero', 'RQ', 'SQR', 'VA'))
* Valor de epsilon para a raiz quadrada dada na aula: 0.0000001