Week 04 - Some ideas and exercises

Aula prática nº 4 – Funções

Simple to warm up

1) Crie uma função, `IMC(peso, altura)`, para calcular o índice de massa corporal, $IMC = \frac{peso}{altura^2}$, dados o peso (em kg) e a altura (em metros). Use-a num programa que peça esses dados ao utilizador.

4) Crie uma função que devolva o maior dos seus dois parâmetros. Por exemplo, `max2(4,-5)` deve devolver 4 enquanto `max2(-3,-2)` deve devolver -2.

5) Use a função anterior para criar uma função `max3` que devolva o maior dos seus 3 parâmetros.

problems

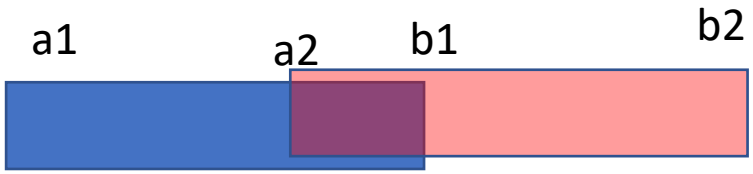
- Intervalos
  - Ajustar intervalo
  - Intersecção de intervalos
- Datas
  - Ano bissexto
  - Proximo dia (i.e. data)
- polinomios
  - Calcular polinomio do Segundo grau
  - Raizes?
- Plot…

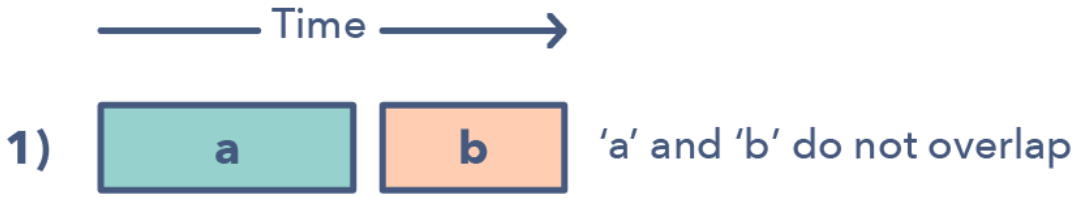
6. Escreva uma função `intersects(a1,b1,a2,b2)` que devolva `True` se os intervalos [a1,b1[e [a2,b2[se intersectarem e devolva `False`, caso contrário. Admita que $a_1 \leq b_1 \wedge a_2 \leq b_2$. *Sugestão: é mais simples definir quando os intervalos não se intersectam.*

Need to select a model

Ways in which (a,b) can overlap with (x,y)

x is between a and b





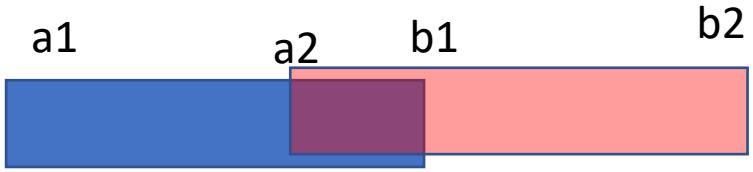
'a' and 'b' do not overlap

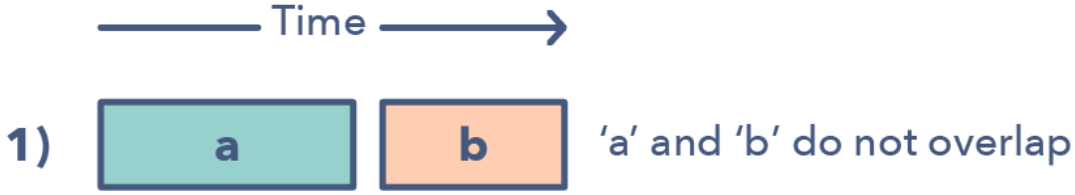
Need to select a model

The model suggested in the exercise

Ways in which (a,b) can overlap with (x,y)

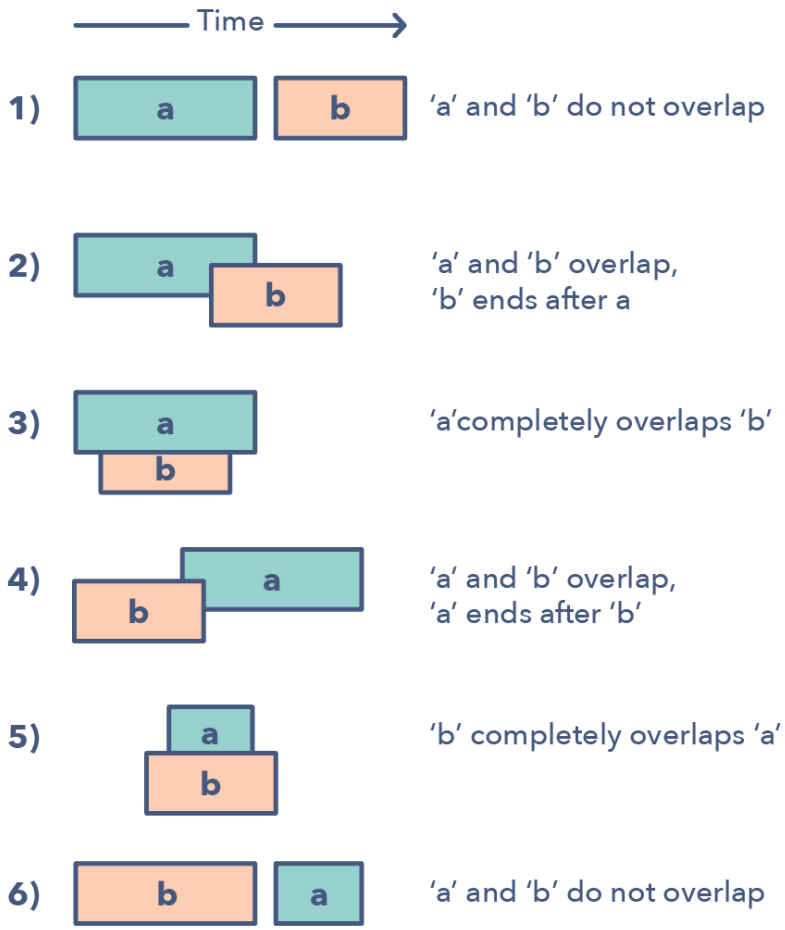
x is between a and b





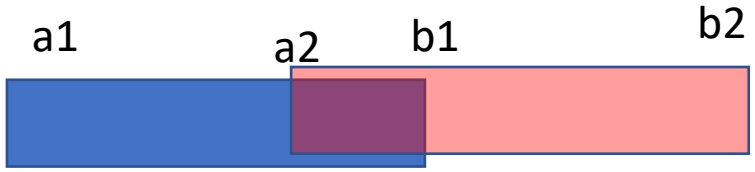
'a' and 'b' do not overlap

What the situations we need to consider?



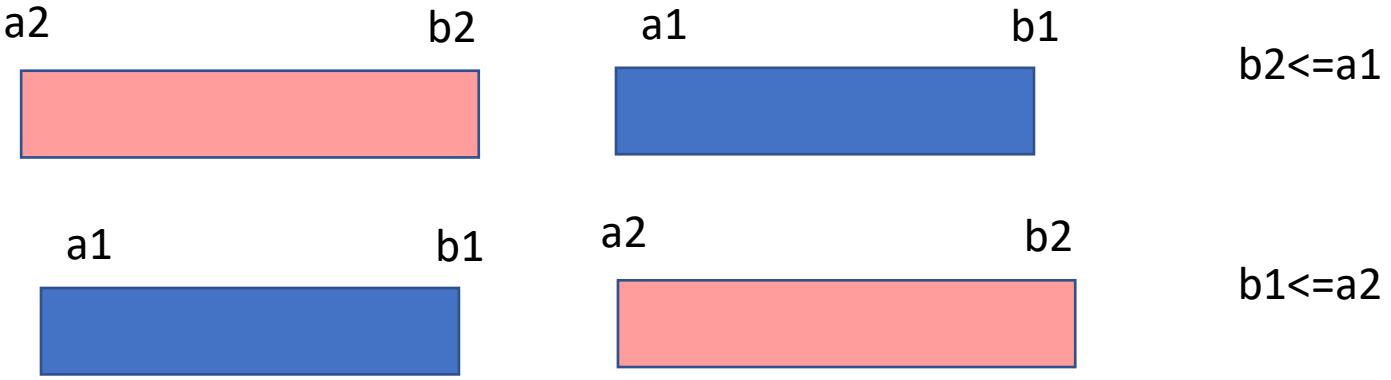
What the situations we need to consider?

The model suggested in the exercise



as $I_a = [a_s, a_e]$ and $I_b = [b_s, b_e]$, while the output interval is defined as $I_o = [o_s, o_e]$. We can find the intersection $I_o = I_a \bigcap I_b$ doing the following:

if ($b_s > a_e$ or $a_s > b_e$) { **return** $\emptyset$ }



return not(a1 >= b2 or a2 >= b1)

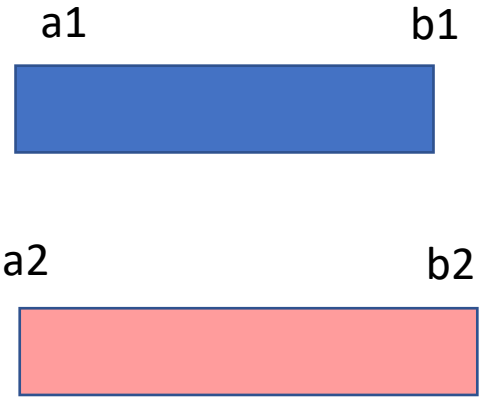
What the function will do??

- Receive two intervals
- Return?
  - A Boolean – True is there is an intersection, False otherwise

```
#receive to intervals [] and [] and
# return True is there is intersection, False otherwise
def intersect( a,b,a1,b1 ) :
    #if intersects
    # need to complete with the intersection condition
    return False  #otherwise
```

What the function will do??

- Receive two intervals
- Return?
  - A Boolean – True is there is an intersection, False otherwise

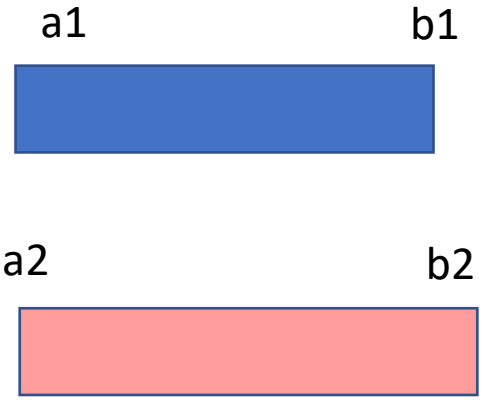



```
#receive to intervals [a1,b1] and [a2,b2] and
# return True is there is intersection, False otherwise
def intersect( a1,b1,a2,b2 ) :

    #if intersects
    # need to complete with the intersection condition
    return not(a1 >= b2 or a2 >= b1)
```

What the function will do??

- Receive two intervals
- Return?
  - A Boolean – True is there is intersection, False otherwise



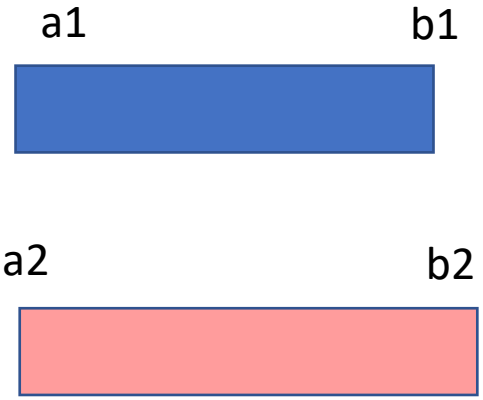

```
#receive to intervals [a1,b1] and [a2,b2] and
# return True is there is intersection, False otherwise
def intersect( a1,b1,a2,b2 ) :

    #if intersects
    # need to complete with the intersection condition
    return not(a1 >= b2 or a2 >= b1)
```

Define parameters and assumptions

What the function will do??

- Receive two intervals
- Return?
  - A Boolean – True is there is intersection, False otherwise



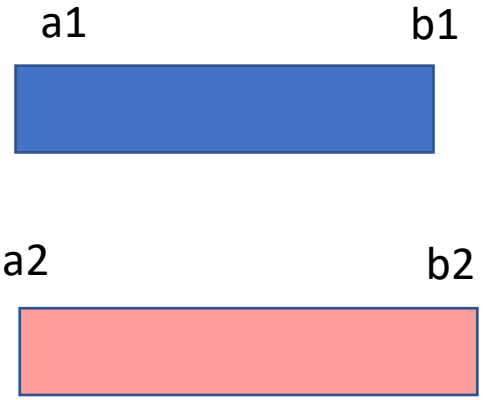

```
#receive to intervals [a1,b1] and [a2,b2] and
# return True is there is intersection, False otherwise
def intersect( a1,b1,a2,b2 ) :

    #if intersects
    # need to complete with the intersection condition
    return not(a1 >= b2 or a2 >= b1)
```

Define results/return and their meaning

What the function will do??

- Receive two intervals
- Return?
  - A Boolean – True is there is intersection, False otherwise




```
#receive to intervals [a1,b1] and [a2,b2] and
# return True is there is intersection, False otherwise
def intersect( a1,b1,a2,b2 ) :

    #if intersects
    # need to complete with the intersection condition
    return not(a1 >= b2 or a2 >= b1)
```

Anos bissextos

7) Analise e execute o programa `dates.py`. A função `isLeapYear` devia indicar quando um ano é bissexto, mas tem um erro. Corrija-a. Um ano é bissexto se for múltiplo de 4, com exceção dos fins de século (múltiplos de 100), que só são bissextos se forem múltiplos de 400. Por exemplo: 1980, 1984, 2004 foram bissextos; 1800, 1900, foram anos comuns, mas 2000 foi bissexto.
8) No mesmo programa, a função para determinar o número de dias de um mês também está errada. Quando o mês é fevereiro, invoque a função anterior para determinar se o ano é bissexto e devolva 29 dias nesse caso.
9) Ainda no mesmo programa, corrija a função `nextDay` para devolver o dia seguinte corretamente.

Look at what you need…

```
def isleap( y ) :
        #complete the code
        # add something to make it work
        return False #assume not leap by default

Def daysMonth( m , y ) :
        #some months have 30 days
        if … complete …
                return 30
        else if m==2  :  # unless they are February
                # if year is leap , return 29
                # else return 28
        return 31 # most month have 31 days, default
```

Ok something is missing? Seen anywhere?

```
Def daysMonth( m , y ) :
        #some months have 30 days
        if … complete …
                return 30
        else if m==2  :  # unless they are February
                # if year is leap , return 29
                # else return 28
        return 31 # most month have 31 days, default
```

Already have isLeap…

```
Def daysMonth( m , y ) :
        #some months have 30 days
        if … complete …
                return 30
        else if m==2   :  # unless they are February
                #if you have function already
                # that solve the problem… use them
                # if year is leap , return 29
                # else return 28
        return 31 # most month have 31 days, default
```

Need it to handle leap years

Why the y parameter?

```
Def daysMonth( m , y ) :
        #some months have 30 days
        if … complete …
                return 30
        else if m==2   :  # unless they are February
                #if you have function already
                # that solve the problem… use them
                # if year is leap , return 29
                # else return 28
        return 31 # most month have 31 days, default
```

Need it to handle leap years

Why the y parameter?

```
Def daysMonth( m , y ) :
        #some months have 30 days
        if … complete …
                return 30
        else if m==2   :  # unless they are February
                #if you have function already
                # that solve the problem… use them
                if isleap( y ) :
                        return 29
                else:
                        return 28
        return 31 # most month have 31 days, default
```

Quadratic formula

The quadratic formula says that

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for any quadratic equation like:

$$ax^2 + bx + c = 0$$

https://www.khanacademy.org/math/algebra/x2f8bb11595b61c86:quadratic-functions-equations/x2f8bb11595b61c86:quadratic-formula-a1/a/discriminant-review

polynome

The quadratic formula says that

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for any quadratic equation like:

$$ax^2 + bx + c = 0$$

- Use coeficients to represent them
  - a, b and c for ax2 + bx + c

- How to calculate p(x) where p(x)=ax2+bx+c
  - Define a function

```
def poly(a,b,c,x):
    return a*x**2 + b*x+ c
```

- How to discriminat of p(x)

```
def discr(a,b,c):
    return b**2 -4*a c
```

Now …

- How to calculate p(3) where a=1,b=2,c=3
- Make a program that
  - Reads the coeficients of a 2nd degree polynome
  - Indicates the number of roots of a polynome
    - Two, one, or none
  - Print out the roots if any

- Note: use the previous functions and the your knowledge on 2nd degree polynomes

Quadratic formula

The quadratic formula says that

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

for any quadratic equat

$$ax^2 + bx + c = 0$$

The discriminant can be positive, zero, or negative, and this determines how many solutions there are to the given quadratic equation.

- A **positive** discriminant indicates that the quadratic has **two distinct real number solutions**.
- A discriminant of **zero** indicates that the quadratic has a **repeated real number solution**.
- A **negative** discriminant indicates that **neither of the solutions are real numbers**.

https://www.khanacademy.org/math/algebra/x2f8bb11595b61c86:quadratic-functions-equations/x2f8bb11595b61c86:quadratic-formula-a1/a/discriminant-review

Do you understand this “draft" solution?

```
#sendo a,b,c coeficientes do polinomio ax2 + bx + c
# a funcao calcula o valor do polinomio para  o valor x
def pol( a , b ,c , x ) :
    return a*x**2 + b*x + c

# a funcao retorna o numero de raizes (nraiz ) e as raizes se existirem
#  nraiz, x1, x2
def root( a,b,c ) :

    discr = ...#  b2 - 4ac complete
    if discr>0 :
        x1 = ... # complete
        x2 = ... # complete
        return 2,x1,x2
    elif discr == 0 :
        x1 = ...
        return 1,x1,x1
    else:
        return 0,0,0
```

Do you understand this “draft" solution?

```
#sendo a,b,c coeficientes do polinomio ax2 + bx + c
# a funcao calcula o valor do polinomio para  o valor x
def pol( a , b ,c , x ) :
    return a*x**2 + b*x + c

# a funcao retorna o numero de raizes (nraiz ) e as raizes se existirem
#  nraiz, x1, x2
def root( a,b,c ) :

    discr = ...#  b2 - 4ac complete
    if discr>0 :
        x1 = ... # complete
        x2 = ... # complete
        return 2,x1,x2
    elif discr == 0 :
        x1 = ...
        return 1,x1,x1
    else:
        return 0,0,0
```

```
a= int( input("a?"))
b= int( input("b?"))
c= int( input("c?"))

nr,x1,x2 = root( a , b ,c )

if nr==2 :
    print("x1=",  x1," x2=", x2)
elif nr==1 :
    print("x1=",  x1 )
else :
    print(" nao tem raizes ")
```

Do you understand this “draft" solution?

```
#sendo a,b,c coeficientes do polinomio ax2 + bx + c
# a funcao calcula o valor do polinomio para  o valor x
def pol( a , b ,c , x ) :
    return a*x**2 + b*x + c

# a funcao retorna o numero de raizes (nraiz ) e as raizes se existirem
#  nraiz, x1, x2
def root( a,b,c ) :

    discr = ...#  b2 - 4ac complete
    if discr>0 :
        x1 = ... # complete
        x2 = ... # complete
        return 2,x1,x2
    elif discr == 0 :
        x1 = ...
        return 1,x1,x1
    else:
        return 0,0,0
```

Complete the missing bits

```
a= int( input("a?"))
b= int( input("b?"))
c= int( input("c?"))

nr,x1,x2 = root( a , b ,c )

if nr==2 :
    print("x1=",  x1," x2=", x2)
elif nr==1 :
    print("x1=",  x1 )
else :
    print(" nao tem raizes ")
```

factorial

$U_0 = 1$ se $n=0$

$U_{n+1} = n * U_n$ se $n>0$

```
def factorial( n ):
    if n=0 :
       return 1
     else:
       return  n * factorial( n-1)
```

Countdown(n)

Se n = 0 então cheguei ao fim - caso base

Senão tenho de contar n-1 i.e. countdow(n-1)

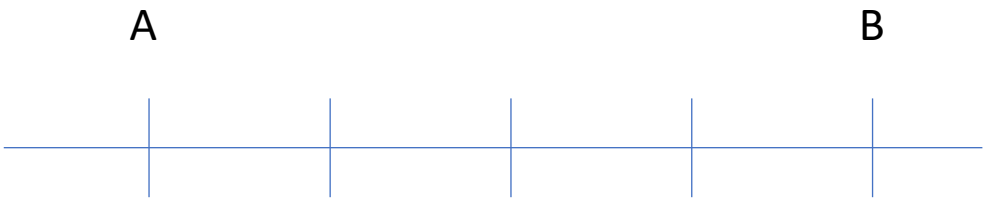
Countdown(n)

Se n = 0 então cheguei ao fim - caso base

Senão tenho de contar n-1 i.e. countdow(n-1)

```
def countdown( n ) :   # definir a função
      if n=0 :  # terminei….
        return True
      else: # tenho de contar n-1 até 0
            print( n )  # conto o ‘n’
            return countdown( n-1)
```

Challenges (*)



- Plot – from week01
  - Change the original program to:
    - Read the interval for the plot a,b
    - Read the number of intervals
    - Replace the function arrange with a function defined by you

```
import numpy as np
import matplotlib.pyplot as plt

plt.figure(1)

t = np.arange(-2.0, 10.0, 0.1)   # try printing t

# print(t)
```

(*) Only if you already talked about list

The end