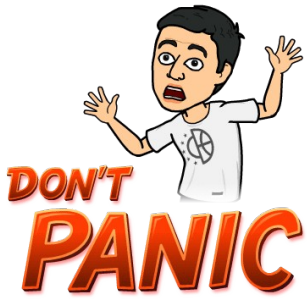
Lista de Exercícios – Repetição
Prof: Yuri Frota

1) Interpretar e traduzir para Python a sequência de comandos em Português a seguir:



Algoritmo {escrita dos termos de Fibonacci menores que L}
leia o valor L
{Processamento dos dois primeiros termos}
Atribua o valor 1 ao termo1
se ele for menor do que L então
  escreva-o
fim se
Atribua o valor 1 ao termo2
se ele for menor do que L então
  escreva-o
fim se
{Processamento dos termos restantes}
enquanto novo termo1 mais termo2 for menor ou igual a L faça
  Calcule o novo termo somando os 2 anteriores
  escreva o termo
  Atribua termo2 a termo1
  Atribua termo a termo2
fim enquanto
Fim algoritmo.

R:

```
L = int( input("Digite o limite L para a sequencia de Fibonacci: ") )

n1 = 1
if n1 < L :
  print(n1)

n2 = 1
if n2 < L :
  print(n2)

while (n1 + n2 <= L) :
  n3 = n1 + n2
  print(n3)
  n1 = n2
  n2 = n3
```

2) Faça um programa em Python que:

a) Escreva um programa que permita que o usuário indique um número de inteiros “n” a serem lidos (entre 1 e 30). Após a leitura dos “n” números, escreva na tela a média, a soma, o produto, o menor valor e o maior valor.

R:

```
nread = int( input("Digite um inteiro entre 1 e 30 : ") )
nsum = 0 # soma dos valores a serem lidos
nprod = 1 # produto dos valores a serem lidos
nmin = -1 # menor valor lido
nmax = 0 # maior valor lido

# faz nread iteracoes (i de 0 a nread-1)
for i in range(0, nread) :
  num = int(input())
  nsum = nsum + num
  nprod = nprod * num
  if nmin == -1 or num < nmin :
    nmin = num
  if num > nmax :
    nmax = num

print("media: %d" % (nsum/nread))
print("soma: %d" % nsum)
print("produto: %d" % nprod)
print("min: %d" % nmin)
print("max: %d" % nmax)
```

b) Faça um programa em Fortran para construir a tabela de multiplicação de números de 1 a 10 (ex.: 1 x 1 = 1, 1 x 2 = 2, ...,2 x 1 = 2, 2 x 2 = 4, ..., etc.).

R:

```
for i in range(1,11) :
  for j in range(1,11) :
    print("%d x %d = %d" % (i, j, i*j) )
```

c) gerar os cinqüenta primeiros termos da série: 1 + N, 5 * N, 9 + N, 13 * N, ..., onde N é um valor lido.

R:

```
N = int(input("Digite um inteiro: "))
n = 1 # numero em {1,5,9,13,...}
for i in range(1,51) :
  if (i%2 == 0) : # se i for par, multiplica
    print(n * N)
  else : # senão, soma
    print(n + N)
  n = n + 4
```

d) determinar todos os números de 3 algarismos, cujas somas dos cubos dos algarismos sejam iguais ao próprio número. Exemplo: 153 = 1**3 + 5**3 + 3**3

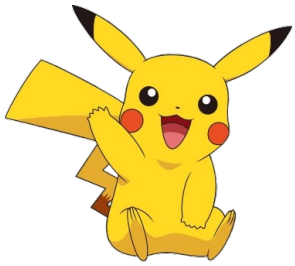
R:

```
for x in range(100,1000) :
  dig1 = int(x/100) # obtem primeiro digito
  dig2 = int((x%100)/10) # obtem segundo digito
  dig3 = x%10 # obtem terceiro digito
  if(dig1**3 + dig2**3 + dig3**3 == x) :
    print(x)
```

e) determinar todos os números de 4 algarismos que possam ser separados em dois números de dois algarismos que somados e elevando-se a soma ao quadrado obtenha-se o próprio número. Exemplo: 3025 = (30 + 25) = 55, e 55**2 = 3025 .

R:

```
for x in range(1000,10000) :
  part1 = int(x/100) # obtem primeiro numero
  part2 = x%100 # obtem segundo numero
  soma = part1 + part2;
  if(soma**2 == x) :
    print(x)
```



f) Suponha que um jogador A de PokemonGO tenha 800 pokemons com uma taxa anual de crescimento/captura de 3% e que o jogador B tem 2000 pokemons com uma taxa de crescimento/captura de 1.5%. Faça um programa que calcule e retorne o número de anos necessários para que o jogador A ultrapasse ou iguale o número de pokemons do jogador B, mantidas as taxas de crescimento.

R:

```
#coding: utf-8


qtdPokemon_A = 800

qtdPokemon_B = 2000

nYear = 0

while qtdPokemon_A < qtdPokemon_B :

  qtdPokemon_A = qtdPokemon_A + int(0.03 * qtdPokemon_A)

  qtdPokemon_B = qtdPokemon_B + int(0.015 * qtdPokemon_B)

  nYear = nYear + 1

str1 =  """

Em %d anos, o jogador A terá pelo menos a mesma quantidade de pokemón
que B!

               .-. \_/ .-.
               \.-\/=\/.-/
             '-./___|=|___\.-'
            .--| \|/`"`\|/ |--.
           (((_)\  .---.  /(_)))
            `\ \_`-.   .-'_/ /`_
              '.__       __.'(_))
                  /     \     //
                 |       |__.'/
                 \       /--'`
             .--,-' .--. '----.
            '----`--'  '--`----'

"""

print(str1 % nYear)
```

3) Escreva um programa para gerar dois valores aleatórios inteiros “x” e “y” entre 1 e 100, que representam o poder e a resistencia de uma carta de magic (para gerar o número aleatório usar **randint**). Após isso, deve-se gerar a seguinte mensagem: “quanto é o poder x multiplicado pela resistencia y da carta ?”, substituindo os números gerados por “x” e “y”. Depois da mensagem, deve ser lida uma resposta do teclado e deve ser exibido uma mensagem indicando acerto ou erro. O programa deve implementar um laço que obrigue o jogador a acertar pelo menos três vezes a resposta antes de sair. Ao final devem ser indicado o número de tentativas, de acertos e de erros.

R:

```
import random

nAcertos = 0
nErros = 0

while nAcertos < 3 :
  poder = random.randint(1,100)
  resistencia = random.randint(1,100)
  print("Quanto é o poder %d multiplicado pela resistencia %d da carta
?" % (poder,resistencia) )
  resposta = int(input())
  if resposta == poder * resistencia :
    print("Acertou!")
    nAcertos = nAcertos + 1
  else :
    print("Errou!")
    nErros = nErros + 1

print("Numero de tentativas: %d" % (nAcertos+nErros) )
print("Numero de acertos: %d" % nAcertos)
print("Numero de erros: %d" % nErros)
```

4) Faça um programa que determina se dois valores inteiros e positivos A e B são "**Bros**" (dois números inteiros são ditos "Bros", caso não exista divisor comum aos dois números).

R:

O método de Euclides é um dos algoritmos mais antigos (300 a.C.) e um dos mais eficientes para calcular o Máximo Divisor Comum (M.D.C) de dois números inteiros diferentes de zero.

O algoritmo se baseia na seguinte propriedade:

MDC(A,B) = MDC(B, A%B)

que deve ser explorada iterativamente até que A%B seja 0 e B seja considerado o MDC. Por exemplo, MDC(252,105) = MDC(105,42) = MDC(42,21) = 21, pois 42%21 é igual a zero. Portanto MDC(252,105) = 21.

```
A = int(input("Digite A: "))
B = int(input("Digite B: "))

if B > A : # ordena par de numeros
  aux = A
  A = B
  B = aux

while (B != 0) :
  r = A % B
  A = B
  B = r

if A == 1 :
  print("São Bros!")
else :
```

```
print("Não são Bros! M.D.C. : %d" % A)
```