```
# e-fólio A
# Compilação 2023/24
# Linguagem: MONTy Python
# (My Own Narrowed Typed Python)
# Compilador da linguagem: FCC
# (descobrir o que quer dizer FCC vale 0,1 na nota final)
# Neste texto tem tudo o que precisa de saber sobre a linguagem
# Se mesmo assim houver dúvidas, ver nota final
# Os comentários já sabem, começam com # e ignora-se tudo daí até ao fim de linha
# Vamos agora às variáveis
# Contrariamente ao habitual no Python, as variáveis têm tipos declarados explicitamente
# Os tipos simples são float e int
# Se nada for declarado, a variável é do tipo float
x = 1          # x é do tipo float, se não tiver sido declarada antes
int x = 1      # x é do tipo int
float x = 1    # x é do tipo float
int x = 1.5    # ERRO: atribuição de valor não inteiro a uma variável inteira na declaração
# um inteiro é representado por um ou mais dígitos de 0 a 9, com ou sem sinal + ou -
```

```
# um float tem 3 representações possíveis
# tal como o inteiro, um ou mais dígitos de 0 a 9, com ou sem sinal + ou -
# (exemplos: 5, 0, -3, +2)
# um número com ponto a separar a parte inteira da parte decimal,
# tendo sempre pelo menos um dígito à esquerda e outro à direita do .
# com ou sem sinal
# (exemplos: 1.0, 0.1, 0.0, 000000.000000, -3.0, +1.5)
# um número em notação científica, composto por um número inteiro ou com casas decimais
# seguido da letra e ou E e de um número inteiro
# (exemplos: 3e2, -2.3e+5, +2.0E-2)
# se tiver uma operação aritmética ou condicional entre valores ou variáveis de tipos diferentes
# o inteiro é convertido para float automaticamente
# assim, dada uma operação aritmética ou condicional
# o resultado da operação é:
# do tipo inteiro se os dois operandos são inteiros
# do tipo float se um dos operandos é do tipo float
# exemplos
int i,j=0,k
```

```
# sim, pode-se declarar várias variáveis e iniciar ou não qualquer uma delas
# quando não é iniciada, assume o valor 0
float x,y,z
# aproveitamos para introduzir a atribuição
z = x + y
k = i + j
# sem problemas, não há alteração de tipos durante as operações
z = x * i # valor de i convertido para float antes da multiplicação
# MUITO IMPORTANTE: a variável i continua inteira, o seu valor é que é convertido
z = i + j # o resultado da operação é inteiro, mas é convertido para float antes da atribuição
i = i * x
# este é o único caso em que há conversão de float para int
# o resultado da operação é do tipo float, mas este é convertido para int antes da atribuição
# caso especial da divisão
# / é a divisão com casas decimais, // é a divisão inteira
# quando / é usado, os valores dos operandos são convertidos para float
# quando // é usado, os dois operandos têm de OBRIGATORIAMENTE ser inteiros, ou dá erro
# o mesmo acontece com o operador % (resto da divisão inteira)
```

```
# só pode ser usado entre dois operadores inteiros
# os operadores possíveis são: +, -, *, /, **, // e %
# com o significado habitual do Python
# os operadores condicionais podem ser operadores de comparação <, <=, >, >=, !=, ==
# e operadores lógicos and, or e not
# assim, uma condição será dada por uma expressão condicional, que pode ser:
# uma comparação entre duas expressões aritméticas
# (exemplos: x < 5, x+3*2>=5+y, x==0, x!=0)
# sobre os quais podem ser aplicados operadores lógicos
# (exemplos: x>5 and y<5, x!=0 or y!=0, not (x==0 and y==0))
# Temos ainda listas, que podem ter elementos de um tipo ou de outro
int[] x        # lista de inteiros
float[] x      # lista de floats
# Atribuição
x = [1,2,3]
# se x já foi declarado, tem o tipo da declaração
# caso contrário, é uma lista de floats
# x[i] índice da lista
```

```
# i=0 é o primeiro da lista, 1 o segundo, etc...
# i=-1 é o último da lista, -2 o penúltimo, etc...
# existem três funções especiais sobre listas:
# size(x) devolve o tamanho da lista
# add(x,i) adiciona um elemento na posição i - se k é size(x), add(x,k+1) adiciona no fim
# remove(x,i) remove o elemento na posição i
# ERRO, se posição não existe ou não é possível
# se não iniciada ou vazia, a lista tem tamanho 0
# Passemos às funções
# Nesta linguagem apenas um valor é retornado
# pode ser int, float ou uma lista
# se nada for dito, o tipo de retorno é float
# o mesmo acontece para os argumentos
def f(x,y): # f tem dois argumentos do tipo float e retorna um valor do tipo float
def float f(float x, float y): igual, mas com declaração explícita
def f(int x,y): # neste caso, o primeiro argumento é do tipo int, o resto é float
def int[] f(float[]x,int y):
# retorna lista de inteiros, tem como argumentos uma lista de floats e um inteiro
```

```
# As strings só existem como valores literais, para as funções input e print
x = input(“Introduza um valor inteiro”)
print(“O resultado é “,x)
# a instrução for pode ser usada de duas formas
for i in list:           # para cada elemento de uma lista
for i in range(k):       # i toma os valores de 0 até k-1
for i in range(j,k):     # i toma os valores de j até k-1
for i in range(j,k,n):   # i toma os valores de j até k-1 com incrementos de n
# a instrução while usa uma condição que deve ser satisfeita para permanecermos no ciclo
while z>y:
  ...
# o mesmo para o if e eventuais elif
if x<0:
  ...
elif y<0:
  ...
# qualquer destas 3 instruções pode ter um else: opcional
# as indentações são usadas como no Python
```

NOTA FINAL: Qualquer dúvida adicional deve ser esclarecida no fórum

se for pertinente, este documento acrescentará ou corrigirá partes do texto

as alterações ficarão visíveis com cor vermelha