Ciências ULisboa Faculdade de Ciências da Universidade de Lisboa
DEPARTAMENTO·DE·INFORMÁTICA

Programação 1
Exercícios e Problemas

Folha 3

(ciclos while)

Exercícios

1. Descreva o que faz este programa em Python:

```
n = 1
while n <= 10:
    print(n)
    n = n + 1
```

2. Indique os erros sintáticos no seguinte programa em Python:

```
x = 1
y = 1
while x = 1 and y < 5
    y = y + 2
```

3. Este programa em Python tem como objectivo escrever a tabuada do número inteiro dado pelo utilizador. Explique porque é que este programa não termina, corrija o erro e especifique qual o valor das variáveis `n` e `i` no final da execução do programa corrigido:

```
n = int(input("Escreve um número inteiro: "))
print("Tabuada do", n, ":")
i = 1
while i <= 10:
    print(n, "x", i, "=", n * i)
    i + 1
```

4. Considere este programa em Python:

```
p = "IlovePython"
x = ""
while x != p:
    x = input("Adivinha a palavra chave: ")
print("Muito bem, acertaste!")
```

a) Descreva o que faz este programa.

b) Altere o programa de forma a usar uma nova variável `t` que guarda o número de tentativas feitas pelo utilizador até acertar na palavra chave correcta.

5. Escreva em linguagem Python um programa que leia 10 números inteiros e escreva no ecrã a soma dos que são pares e a soma dos que são ímpares. Caso não seja lido nenhum número par, deve ser indicada a soma 0 para os pares. Idem para os ímpares. *Note que, assim, não se distingue o caso em que a soma dos pares dá* 0 *do caso em que não ocorre nenhum par. Idem para os ímpares.*

6. Considere este programa em Python:

```
dividendo = int(input("Dividendo: "))
divisor = int(input("Divisor: "))
resto = dividendo
quociente = 0
while resto >= divisor:
    resto = resto - divisor
    quociente = quociente + 1
print("O quociente é", quociente, "e o resto é", resto)
```

a) O que faz este programa?

b) Quais são as duas operações aritméticas disponíveis em Python que implementam a mesma funcionalidade?

c) Altere o programa de forma a que, caso o dividendo seja menor que o divisor, o utilizador seja alertado e lhe sejam pedidos novos valores.

d) Faça uma alteração adicional de forma a garantir também que tanto o dividendo como o divisor são positivos.

7. Usando como inspiração o exercício anterior, escreva um programa em Python que peça dois inteiros positivos ao utilizador e devolva a sua multiplicação, usando apenas somas para efectuar o cálculo.

8. Este programa em Python tem como objetivo escrever todos os pares ordenados (i, j) que podem ser formados com os algarismos 0 a 3 ($i,j=0..3$). No entanto, o programa está incompleto pois falta-lhe as inicializações das variáveis `i` e `j`.

```
while i <= 3:
    while j <= 3:
        print("(" + str(i) + "," + str(j) + ")")
        j = j + 1
    i = i + 1
```

a) Acrescente duas linhas de código ao programa, referentes às inicializações de `i` e `j`.

b) Altere o programa de modo a que não escreva os pares formados por algarismos iguais (i,i).

c) Altere o programa de modo a que também não escreva os pares tais que o par inverso já foi escrito, isto é, se (i,j) já foi escrito então (j,i) não deverá ser escrito.

d) Introduza no programa uma nova variável que guarda o número total de pares que foi escrito.

9. O seguinte programa em Python escreve no ecrã os factoriais de todos os números inteiros entre 1 e `n`, em que `n` é dado pelo utilizador:

```
n = int(input("Escreve um número inteiro: "))
current_n = 1
while current_n <= n:
    i = current_n
    f = 1
    while i > 1:
        f = f * i
        i = i - 1
    print("Factorial de " + str(current_n) + ": " + str(f))
    current_n = current_n + 1
```

Altere o programa de forma a que não escreva os factoriais cujo valor é superior a 1000.

10. Considere o seguinte excerto de código em Python:

```
while x >= 1:
    x = x - 1
    if x == 1:
        x = 0
    else:
        x = 1
```

a) Estude o que acontece durante a execução, para diferentes valores iniciais de `x`. Em particular, descubra quais os dois casos (sim, dois!) em que `x` tem o valor `0` no final da execução.

b) Remova as duas linhas de código referentes ao `else` e estude de que forma o comportamento do programa é alterado. Retire também as duas linhas de código referentes ao `if` e verifique se o comportamento é alterado.

11. Considere o seguinte excerto de código em Python onde, para diferentes valores iniciais de `x`, o ciclo `while` termina por motivos diferentes:

```
y = 1
while x >= 1 and y < 3 and x != y:
    x = x - 1
    y = y * -1 + 1
```

a) Inicializando `x` com o valor 6, verifique quais são os valores de `x` e `y` de cada vez que o ciclo é executado. Descreva o comportamento de ambas as variáveis.

b) Inicialize `x` com diferentes valores à sua escolha e para cada caso verifique qual das condições da guarda do `while` é responsável pela terminação do ciclo. Alguma das condições poderia ser removida sem afetar a funcionalidade do código?

c) Estude o comportamento do código para valores iniciais de `y` diferentes de 1.

Problemas

Ciclos que podem ser infinitos, em programas com interacção.

1. Escreva um programa em Python que repetidamente peça ao utilizador um número inteiro e escreva no ecrã o seu quadrado. O programa deve terminar quando o utilizador introduzir o número 0.

2. Escreva um programa em Python que repetidamente peça ao utilizador um número decimal, escreva no ecrã a sua parte inteira, e pergunte em seguida se o utilizador quer continuar a execução do programa.

Padrão listagem (até limite dado)

3. Escreva um programa em Python que peça ao utilizador um número inteiro positivo n e escreva no ecrã os números inteiros de 0 a n (inclusive), um por linha.

4. Escreva um programa em Python que peça ao utilizador um número inteiro positivo n e escreva no ecrã todas as potências de 2 com expoentes entre 0 e n (inclusive), uma por linha.

5. Escreva um programa em Python que peça ao utilizador um número inteiro positivo k e uma palavra w, e escreva no ecrã k linhas em que cada linha i tem o número da linha i separado, por um espaço, da palavra w concatenada i vezes. Note que i está no intervalo de 1 a k (inclusive).

6. Escreva um programa em Python que peça ao utilizador um número inteiro não-negativo n e escreva no ecrã todas as potências de 2 com valores entre 0 e n (inclusive), uma por linha.

Padrão acumulador

7. Escreva um programa em Python que peça ao utilizador um número inteiro positivo k e escreva no ecrã o resultado do somatório dos primeiros k inteiros positivos.

8. Escreva um programa em Python que peça ao utilizador um número inteiro positivo k e escreva no ecrã o resultado do produtório das potências de 3 com expoente de 0 a k inclusive.

Padrão contador (com filtro)

9. Escreva um programa em Python que peça ao utilizador um número inteiro não-negativo k e escreva no ecrã o número de múltiplos de 3 no intervalo de 0 a k inclusive.

10. Escreva um programa em Python que peça ao utilizador um número inteiro não-negativo k e escreva no ecrã o número de ímpares no intervalo de 0 a k inclusive.

11. O João encontra-se no quinto degrau de uma escadaria (a contar de baixo) e começa a subir os degraus dois a dois. A Joana encontra-se no vigésimo degrau da mesma escadaria e ao mesmo tempo começa a descer os degraus um a um. Escreva um programa em Python que diga em que degrau se encontra cada um deles imediatamente antes de se encontrarem/cruzarem, e que indique também se eles se vão encontrar no mesmo degrau. Altere o programa de forma a que seja o utilizador a indicar quais os degraus de partida de ambos.

Padrão acumulador com filtro

12. Escreva um programa em Python que que peça ao utilizador um número inteiro positivo k e escreva a soma dos seus divisores próprios. Um divisor é próprio se for diferente do número e da unidade. Por exemplo a soma dos divisores próprios de 12 é 2 + 3 + 4 + 6 = 15.

13. Designa-se por números perfeitos os números que são iguais à soma dos seus divisores próprios mais um. Escreva um programa em Python que peça ao utilizador um número inteiro positivo k e escreva no ecrã se k é perfeito. Por exemplo, 496 é perfeito porque é igual à soma dos seus divisores próprios (2; 4; 8; 16; 31; 62; 124; 248) mais 1.

Padrão acumulador e contador

14. (a) Escreva um programa em Python que peça ao utilizador um número inteiro positivo k, e escreva no ecrã o número de vezes que se consegue dividir *exactamente* k por 2.
(b) Escreva um programa em Python que peça ao utilizador um número inteiro positivo k, e escreva no ecrã a parte inteira do logaritmo de k na base 2.

Padrão acumulador e contador: ciclos encaixados

15. Escreva um programa em Python que peça ao utilizador um número inteiro positivo k maior do que 2 e escreva no ecrã quantos números perfeitos existem entre 2 e k (inclusive). Por exemplo, existem 4 números perfeitos entre 2 e 10000 (o 6, o 28, o 496 e o 8128).

Padrão busca por enumeração

16. Escreva um programa em Python que peça ao utilizador um número inteiro k e escreva no ecrã se k é ou não um cubo perfeito.

17. Designa-se por número primo um número maior do que 1 que não tem divisores próprios. Escreva um programa em Python que que peça ao utilizador um número inteiro não-negativo k e que escreva no ecrã se k é primo. Por exemplo, 2, 3, 5, 7 e 7919 são números primos.

18. Escreva um programa em Python que "adivinhe" o número inteiro entre 0 e 100 em que o utilizador está a pensar, através de uma sucessão de tentativas às quais o utilizador deve responder `maior` (ou `+`), `menor` (ou `-`) ou `certo`. Exemplo de interação com o computador (em itálico os dados introduzidos pelo utilizador):

```
Pense num número inteiro entre 0 e 100.  (Enter/Return)
O número é 50.
maior
O número é 76.
menor
O número é 63.
enor
Resposta inválida.
O número é 63.
-
O número é 57.
+
O número é 60.
certo
Yupiiii!!
```

Padrão busca por enumeração e contador: ciclos encaixados

19. Escreva um programa em Python que peça ao utilizador um número inteiro k maior do que 2 e escreva no ecrã quantos números primos existem entre 2 e k (inclusive). Por exemplo, existem 1000 números primos entre 2 e 7919.

20. Escreva um programa em Python que peça ao utilizador um número inteiro k maior do que 10 e escreva no ecrã quantas capícuas existem entre 10 e k. Uma capícua é um número que se lê de igual forma da esquerda para a direita e da direita para a esquerda. Por exemplo, entre 10 e 100 existem 9 capícuas.