Iteração e laços

Fernando Castor

Vamos construir um jogo de adivinhar o número. O jogo tem dois jogadores. O **Jogador 1** indica um número a ser advinhado. Depois disso, esse número some da tela e o **Jogador 2** tem três chances para adivinhar o número escolhido. Para cada tentativa, o jogo deve dizer ao jogador se o número que ele forneceu é maior, menor ou igual ao correto.

*Pyt

File Edit Format Run Options W

```
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
print()
```

Como remover toda essa repetição?

Laços (ou *loops*)

Executam um comando enquanto uma condição for verdadeira

Similares a comandos `if`, só que executam várias vezes

```
...
print()
print()
print()
print()
print()
print()
print()
print()

palpite = input("Dê um palpite.")
palpite = int(palpite)
...
```

Forma geral do comando

`while`

```
while condição:
    comandos1

comandos2
```

```
...
print()
print()
print()
print()
print()
print()
print()
print()

palpite = input("Dê um palpite.")
palpite = int(palpite)
...
```

```
numLinhas = 50

while numLinhas > 0 :
  print()
  numLinhas = numLinhas - 1

palpite = input("Dê um palpite.")
palpite = int(palpite)
...
```

Condição do laço. Parte mais importante!

```
numLinhas = 50

while (numLinhas > 0) :
   print()
   numLinhas = numLinhas - 1

palpite = input("Dê um palpite.")
palpite = int(palpite)
...
```

Crie um programa que lê um número *N* do teclado e calcula o seu fatorial.

```
n = input("Calcular o fatorial de qual número?")

# transforma o string lido em um número
n = int(n)
fat = 1

while (n > 1) :
   fat = fat * n
   n = n - 1

print("O valor do fatorial é", str(fat))
```

Voltando ao código do programa de adivinhar o número

Faça com que a parte que pede os palpites e checa se estão corretos também rode em um laço

Depois disso, modifique o programa para que peça 10 palpites, ao invés de 3

Faça um programa que, dado um número lido a partir do teclado, calcula a soma de seus dígitos (independentemente do número de dígitos)

Voltando ao jogo de adivinhar o número, modifique-o para que, uma vez que o jogador acerte o número, o laço termine imediatamente, independentemente do número máximo de palpites

```
while numPalpites < limitePalpites) :
  if not(acertou):
    print("Dê um palpite.")
    palpite = int(input("Dê um palpite: "))

    if palpite < numeroAAdivinhar :
      print("Seu palpite é menor...")
    elif palpite > numeroAAdivinhar :
      print("Seu palpite é maior…")
    eles :
      print("ACERTOU.")
      acertou = True
      break

  numPalpites = numPalpites + 1
```

```
...

while (not(acertou) & (numPalpites < limitePalpites)):
  if not(acertou):
    palpite = int(input("Dê um palpite."))

    ...
```

Faça com que seu programa só aceite números a ser adivinhados entre 0 e 100. Enquanto os números não estiverem nesse intervalo, seu jogo deve continuar repetidamente pedindo que o número a ser adivinhado seja fornecido.

Construa um programa que lê um número inteiro *N* a partir do teclado e calcula a raiz cúbica inteira desse número, se houver. Caso não haja, ele deve informar isso ao usuário.

A sequência de Fibonacci é dada por

fib(0) = 1
fib(1) = 1
fib(n) = *fib*(n-1) + *fib*(n-2)

Faça um programa que, dado um número n, calcula o *fib*(n).

Faça um programa que, dado um raio ***R***, desenha apenas usando asteriscos um círculo cujo raio é ***R*** caracteres.

Construa um programa que lê um número *X* a partir do teclado e verifica se ele é primo. Seu programa deve informar ao usuário se *X* é primo ou não.

```
numero = int(input("Forneca um numero inteiro maior que 1: "))

i = 2

if numero == 2 :
  print ("O numero 2 eh primo.")
if numero < 2 :
  print ("Numero invalido. Da proxima vez forneca um maior que 1")
else :
  while i < numero :
    if numero % i == 0:
      print("O numero", numero, "nao eh primo. Eh divisivel por ", i)
      break
    else :
      i = i + 1

if i == numero :
  print("O numero", numero, "eh primo.")
```

Escreva um programa que pede que o usuário forneça um número inteiro *Y* e que imprime um par de inteiros, root e pwr, tais que 0 < pwr < 6 e root**pwr é igual a *Y*. Se esse par não existir, seu programa deve informar isso ao usuário.

Fonte: John Guttag, Introduction to Computation and Programming Using Python. MIT Press, 2013.

Construa uma variante do jogo de adivinhar o número que funciona da seguinte maneira. O número inteiro a ser adivinhado ***N*** pode estar **entre 1 e 200**. Palpites podem ter apenas três valores: 1, 10 e 100 (o jogo não deve aceitar valores diferentes). Além disso, o jogo mantém uma variável ***S*** inicialmente igual a 0. Quando o segundo jogador fornece um palpite, um dos dois cenários ocorre:

(1) ***S*** é menor que o número a ser adivinhado e o novo ***S*** é o resultado de **somar** o novo palpite ao ***S*** antigo
(2) ***S*** é maior que o número a ser adivinhado e o novo ***S*** é o resultado de **subtrair** o novo palpite do ***S*** antigo
Se, após essas operações, o valor de ***S*** for igual ao número a ser adivinhado, o jogo acaba. Caso contrário, prossegue conforme as mesmas regras. O jogo deve contar o número de palpites que foram necessários para acertar e imprimir esse número no final.

Faça seu programa imprimir todos os divisores de um número não-primo diferentes de 1 e dele próprio. Apresente essa informação para o usuário **após** dizer para ele que o número não é primo.

Construa um programa que lê um número *N* a partir do teclado e calcula todos os números primos entre 1 e *N*

A solução usa **laços aninhados**

```
numero = int(input("Forneca um numero inteiro maior que 1: "))


numeroAtual = 2

while numeroAtual <= numero :
  i = 2
  if numeroAtual == 2 :
    print ("O numero 2 eh primo.")
  elif numeroAtual < 2 :
    print ("Numero invalido. Da proxima vez forneca um maior que 1")
  else :
    while i < numeroAtual :
      if numeroAtual % i == 0:
        print("O numero", numeroAtual, "nao eh primo. Eh divisivel por ", i)
        break
      else :
        i = i + 1

    if i == numeroAtual :
      print("O numero", numeroAtual, "eh primo.")

  numeroAtual = numeroAtual + 1
```

Em Python, é possível acessar os caracteres de um string a partir de sua posição:

```
>>> a = 'Fernando'
>>> a[3] # 0 é a posição inicial
'n'
```

Construa um programa que ordena alfabeticamente os caracteres de um string, supondo que esse string usa apenas letras maiúsculas ou apenas letras minúsculas (mas não mistura as duas)

Crie um programa que constrói uma tabela de multiplicação. Por exemplo, para um N = 4, a tabela de multiplicação correspondente é a seguinte:

	1	**2**	**3**	**4**
1	1	2	3	4
2	2	4	6	8
3	3	6	9	12
4	4	8	12	16

Note que, em uma célula qualquer (x,y), o valor armazenado é igual a x * y. Seu programa deve receber o valor de N como entrada e produzir uma tabela como essa, organizada. Não se preocupe, porém, se o valor de N for grande demais para apresentar a tabela de forma arrumada.

Fonte: Ben Stephenson, The Python Workbook, cap. 3. Springer, 2014.