Programação em Python

Tópicos

- Iteração: instruções `while, for, break`

Exercícios

1. Para cada excerto de código abaixo, tente prever quantas iterações vão ser executadas e quais valores a ser impressos. Depois visualize a execução de cada excerto usando o PythonTutor.com.

`n = 10`
`while n > 0:`
`  print(n)`
`  n -= 1`
	`n = 1`
	`while n < 1000:`
	`  print(n)`
	`  n *= 2`

`for n in (1, 2, 5, 10, 20, 50):` `  print(n)`	`for c in "abcdefghijkl":` `  print(c)`
`for n in range(20):` `  print(n)`	`for n in range(50, 0, -5):` `  print(n)`

2. Escreva uma função `factorial(n)` que calcule o fatorial de n, definido por $n! = 1 \times 2 \times 3 \times \cdots \times n$. Teste a função com alguns exemplos:

```
print(factorial(6))
print(factorial(4))
```

Proposta de resolução:

```python
#Exercicio 2
def factorial(n):
    f = 1
    for k in range(2, n+1):
        f *= k
    return f

print(factorial(6))
print(factorial(4))
```

3. Pretende-se calcular a média de uma sequência de valores. Escreva um programa que peça ao utilizador uma sequência de números reais. Para terminar a sequência, o utilizador pressiona ENTER, introduzindo uma linha vazia. Nessa altura, o programa deve mostrar a média dos números introduzidos.

Proposta de resolução:

```python
#Exercicio 3
avg=0
count=0
while True:
    num=input("Introduza um numero real ")
    if not num: break
    avg+=float(num)
    count+=1
print("Done. A média dos números", count, " introduzidos é:", avg/count)
```

4. O seguinte programa mostra uma tabela dos quadrados de quatro números naturais. Implemente-o e experimente.

```
#Exercicio 4
# Quadrados dos primeiros 4 números
print("{:2s} {:2s}".format("n", "n²"))
for n in [1, 2, 3, 4]:
    print("{:2d} {:2d}".format(n, n**2))
```

   a) Modifique o programa para mostrar a tabela para números entre 1 e 20. Use a função `range`.

   b) Acrescente uma coluna para mostrar 2^n. Use a mesma formatação que no exemplo dado.

Proposta de resolução:

```
#Exercicio 4
# Quadrados dos primeiros 4 números
print("{:2s} {:2s} {:2s}".format("n", "n²", "2^n"))
for n in range(21):
    print("{:2d} {:2d} {:2d}".format(n, n**2, 2**n))
```

5. ** Lembra-se da sequência de Fibonacci?

   A sequência de Fibonacci é uma sequência de inteiros na qual cada elemento é igual à soma dos dois anteriores: 0, 1, 1, 2, 3, 5, 8, 13, …, ou seja, cada termo obtém-se como $F_n = F_{n-1} + F_{n-2}$. Os primeiros valores são definidos como $F_0 = 0$ e $F_1 = 1$.

   a) Escreva uma função `Fibonacci(n)` para calcular o n-ésimo número de Fibonacci. *Sugestão: em cada iteração atualize e guarde os dois últimos valores da sequência.*

Proposta de resolução:

1. Resolução mais simples, mas mais avançada:

```python
#""" Exercicio 5
def fibonacci(n):
    if n in {0, 1}:
        return n
    return fibonacci(n - 1) + fibonacci(n - 2)  # recursivo

print([fibonacci(n) for n in range(15)])
```

2. Resolução mais imediata:

```python
def fibonacci(n):
    f1=1
    print(f1)
    f2=1
    print(f2)
    f=f1+f2
    print(f)
    while f<=n:
        f1=f2
        f2=f
        f=f1+f2
        print(f)
    return f
```

6. ** Escreva uma função `isPrime(n)` que devolva `True` se o número n é primo e `False`, caso contrário. Um número N é primo se não tiver divisores além de 1 e de N.

   Relembre a abordagem seguida em aulas anteriores: tente dividir o número por 2, por 3, etc. Se encontrar um divisor exato, então o número não é primo.

   Teste a função fazendo um programa que percorre todos os números entre 1 e 100 e indique para cada um se é primo ou não.

Proposta de resolução:

```python
def isPrime(n):
    d = 2
    if n==1: return False
    while d <= (n-1):
        if n % d == 0:
            return False
        d += 1
    return True

def main():
    # test all numbers from 1 to 39
    for n in range(1, 40):
        print(n, isPrime(n))

# call main function:
main()
```