TÍTULO: Funções em Python

CENÁRIO DE APRENDIZAGEM		
Escola:	***Duração (minutos):***	**90**
Professor(a):	***Idade dos alunos:***	**13**

Ideia Chave:	**Vamos conhecer funções em Python**

Tópicos:

- Os alunos aprofundam sua compreensão do uso de vários softwares e políticas.

Objetivos:

- Os alunos serão capazes de projetar e criar programas que utilizam sub-rotinas, estruturas e tipos de dados apropriados, expressões, variáveis e comandos iterativos e condicionais.
- Linguagens de programação gerais são usadas para criar programas.
- Os alunos compreendem as diferentes maneiras de usar simulações e algoritmos de organização passo a passo para resolver problemas.

Resultados:

- Os alunos criam um jogo, aplicativo ou aplicativo móvel mais complexo que resolve um problema particular de um assunto ou tópico específico.
- Os alunos aprendem a delinear a operação de um programa mais complexo em vários padrões e generalizações.

formas de trabalho:

- trabalho individual
- trabalho em pares
- trabalho de grupo

Métodos:

- apresentação
- discussão
- exercício interativo

ARTICULAÇÃO
Linha de atuação (duração, minutos)
INTRODUÇÃO O professor explica e inicia a discussão com os alunos: Ao arranjar soluções num computador, normalmente começamos por analisar o problema e depois tentamos dividi-lo em partes mais pequenas. Desta forma, passamos a olhar para o problema baseando-nos em soluções para cada parte. Como podemos escrever soluções de pequenos programas dentro de um programa com o qual podemos resolver partes do problema, e a que podemos recorrer várias vezes? Nos programas de computador, cada série de comandos tem um papel a desempenhar. Alguns lidam com os dados input, outros com output, algumas partes formam soluções, outras lidam com os dados de acordo com as instruções. Vários comandos que fazem sentido como um todo podem ser separados como uma parte independente do programa, a que se chama função. Até agora, vimos várias funções integradas, como int(), input(), print(), len()… Estas funções já fazem parte do Python e só as usamos quando precisamos delas. Além do uso das funções integradas, também podemos fazer as nossas próprias funções para tarefas específicas. Geralmente, precisamos de fazer funções para separar partes independentes do programa que tendem a repetir-se. Quando fazemos isso, estamos a aperfeiçoar o programa e este trabalha mais rapidamente. No Python, é necessário definir uma nova função, o que significa que temos de usar comandos que descrevam o que fazem. Depois, temos de lhe dar um nome e uma lista de argumentos, que irá usar. Por exemplo:
PARTE PRINCIPAL Nos programas de computador, cada série de comandos tem um papel a desempenhar. Alguns lidam com os dados input, outros com output, algumas partes formam soluções, outras lidam com os dados de acordo com as instruções. Vários comandos que fazem sentido como um todo podem ser separados como uma parte independente do programa, a que se chama função. Até agora, vimos várias funções integradas, como int(), input(), print(), len()… Estas funções já fazem parte do Python e só as usamos quando precisamos delas. Além do uso das funções integradas, também podemos fazer as nossas próprias funções para tarefas específicas. Geralmente, precisamos de fazer funções para separar partes independentes do programa que tendem a repetir-se. Quando fazemos isso, estamos a aperfeiçoar o programa e este trabalha mais rapidamente. No Python, é necessário definir uma nova função, o que significa que temos de usar comandos que descrevam o que fazem. Depois, temos de lhe dar um nome e uma lista de argumentos, que irá usar. Por exemplo: Por exemplo:

```
def funcion_name (paramethers):
    block_of_commands
    return value
```

Após definirmos a função, ativamo-la no programa bastando para tal utilizar o seu nome. Considerando que as funções usam muitas vezes dados input, temos de dar esse tipo de dados à função se queremos que funcione corretamente. As funções podem ter diferentes parâmetros de entrada que o utilizador usa quando e se forem necessários. Há quatro tipos de funções diferentes no Python:

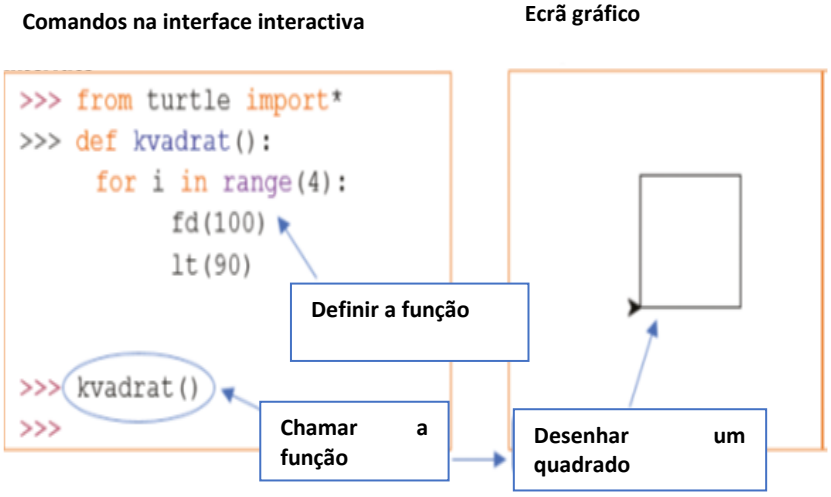
- Funções que não retornam um valor e:
  - Não têm parâmetros de entrada
  - Têm parâmetros de entrada
- Funções que retornam um valor e:
  - Não têm parâmetros de entrada
  - Têm parâmetros de entrada

Uma função que não tem valores de entrada e não retorna valores depois de execução

Esta forma de função não retorna quaisquer valores pelo que não precisamos de usar o comando de retorno. Esta função é executada utilizando o seu nome sem quaisquer parâmetros dentro dos parêntesis: **function_name()**.

EXERCÍCIO 1

Vamos aplicar uma função para desenhar um quadrado com um lado de 100 pixels de lado.



EXERCÍCIO 2

Escreve um programa no qual uses uma função para adicionar dois números dados. Os tamanhos dos números (variáveis a e b) serão definidos fora da função, no programa

Programa de computador **Exemplo de teste**

Computer program

```
a=5
b=10
def zbroji():
    z=a+b
    print('Zbroj je',z)
```

Test example

```
>>> zbroji()
Zbroj je 15
>>>
```

Chamar a função

Neste exercício, escrevemos um programa de computador que adiciona dois valores previamente definidos, e isso é bastante limitador. Uma melhor forma de o fazer poderá ser integrar as variáveis input que queremos adicionar.

Programa de computador

```
def zbroji():
    a=int(input('Broj a? '))
    b=int(input('Broj b? '))
    z=a+b
    print('Zbroj je',z)
```

Exemplo de teste

```
>>> zbroji()
Broj a? 3
Broj b? 4
Zbroj je 7
>>>
```

Se quisermos usar a mesma função de adição várias vezes, podemos usar a **for loop**, para repetir, por exemplo, uma sequência.

Programa de computador

```
def zbroji():
    a=int(input('Broj a? '))
    b=int(input('Broj b? '))
    z=a+b
    print('Zbroj je',z)
for i in range(3):
    zbroji()
    print()
```

Exemplo de teste

```
Broj a? 3
Broj b? 4
Zbroj je 7

Broj a? 5
Broj b? 6
Zbroj je 11

Broj a? 7
Broj b? 8
Zbroj je 15

>>>
```

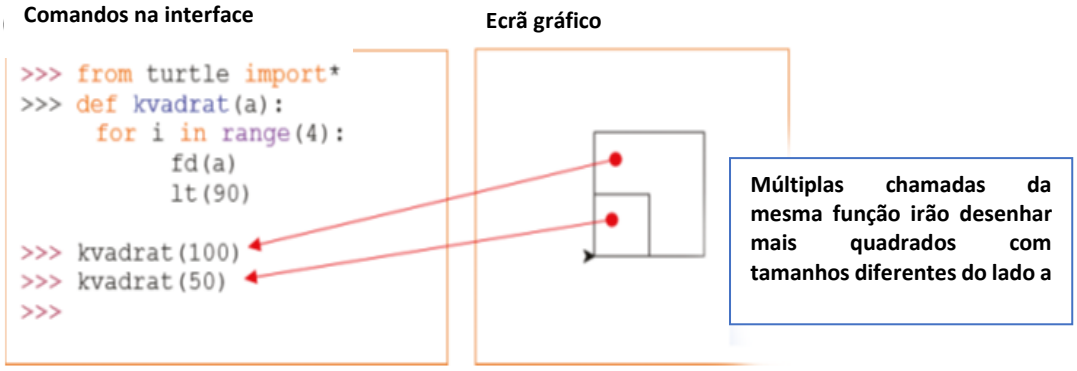
O comando print() tem a tarefa de criar uma linha vazia entre os valores impressos

Uma função que tem valores de entrada e não retorna um valor após a execução

Esta forma de função tem um ou vários parâmetros de entrada e não retorna nenhum valor após a execução. A função é executada chamando o seu nome com uma lista de parâmetros de entrada entre colchetes: **function_name** (lista de parâmetros).

EXERCÍCIO 3

Vamos utilizar uma função que desenhe um quadrado com um pixel de lado. O comprimento do lado "a" é definido ao "chamar" a função como um parâmetro de entrada.

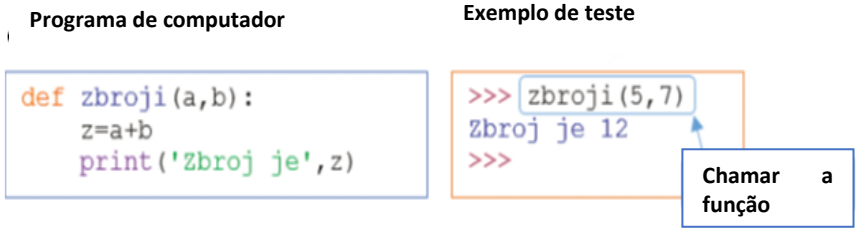


A função **square()** deste exercício usa um parâmetro input escrito entre parêntesis depois do nome da função. Se queremos usar parâmetros input para "chamar" a função, então esses parâmetros têm de ser definidos: **def square(a)**. No Python, uma função é definida usando a palavra-chave **def**.

Uma função pode ter vários parâmetros definidos, que serão mostrados no próximo exercício

EXERCÍCIO 4

Vamos escrever um programa que usará uma função para somar dois números.



Uma função que não tem parâmetros input e não retorna valores depois de executada
Os parâmetros input são escritos quando mudamos a função incorporada **input()**, e o comando "return" dá o resultado do cálculo. O valor que será devolvido ao programa é escrito após o comando de retorno.

EXERCÍCIO 5

Vamos escrever um programa que calcule e imprima uma multiplicação de dois números. Para tal, teremos de usar uma função especial. Os números são escritos para o programa principal como valores input.

```
def multi():
    z*a*b
    return z
a=int(input('Enter the first number: '))
b=int(input('Enter the second number: '))
print('The result is',multi())
```

Explicação:

Usámos vários valores **input** para escrever os valores **a** e **b** e a função **multi()**, que multiplica estes dois números. Depois de executar a função **multi()**, o resultado é armazenado na variável **z**, como resultado de **multi()**, no programa principal.

EXERCÍCIO 6

Escreve um programa que dê entrada a dois números de três dígitos e imprima um número com dígitos menores do que 1. A função escrita irá comparar os dígitos e produzir o número desejado.

```
a=int(input('Write the first number'))
b=int(input('Write the second number'))
a1 = a%0
b1 = b%10
def smaller():
    if a1<b1:
        return a
    else:
        return b
print('Number',smaller(),'has a smaller digit.')
```

Explicação

Ao usar o operador **%** (resto) separámos os dígitos dos elementos escritos em **a** e **b**. Depois armazenámos em **a1** e **b1**. Através da função **smaller()** dentro de **print()** comparámos os valores das variáveis **a1** e **b1** e devolve o valor correspondente ao programa. Como resultado, esse valor será visualizado.

Uma função que tem parâmetros input e retorna valores depois de executada

A mais complicada forma de função é aquela que tem parâmetros input e retorna um valor depois de executada. Podemos ver que essa função tem de ter uma lista de parâmetros expectáveis quando a definimos e também tem de ter um valor depois do comando de retorno.

EXERCÍCIO 7

Vamos escrever um programa que irá calcular a soma dos primeiros números naturais. Dentro do programa, vamos aplicar a função para calcular a adição dos números dados.

Programa de computador

```
def zbroji(n):
    z=0
    for i in range(1,n+1):
        z=z+i
    return z
n=int(input('Upiši broj n: '))
print('Zbroj prvih',n,'brojeva je',zbroji(n))
```

Exemplo de teste

```
Upiši broj n: 5
Zbroj prvih 5 brojeva je 15
>>>
Upiši broj n: 7
Zbroj prvih 7 brojeva je 28
>>>
```

EXERCÍCIO 8

Muitas vezes vemos números primos nos exercícios de matemática. Por exemplo, um estudante tem de verificar se um número é primo ou não. Vamos ajudá-lo a escrever um programa que introduzirá qualquer número como input para verificar se é um número primo. No nosso programa usaremos a função que retornará a mensagem dizendo se um número é primo ou não.

```
def prime(n):
    x='The number is a prime'
    for i in range(2,n):
        if n%i==0:
            x='The number is not a prime'
    return x
n=int(input('Write a number: '))
print(prime(n))
```

Explicação:

Sabemos que os números primos não podem ser divididos exceto pelo um (1) ou pelos próprios. Então só precisamos de verificar quando o **n** pode ser dividido por outros números de 2 a n-1. Se não houver esse número, **n** é um número primo.
A função **for loop** garante que todos os números são verificados. Se encontramos esse número, o valor de **x** será alterado.

EXERCÍCIO 9

Vamos atualizar o nosso último programa para que imprima números até chegar ao zero. Com cada número, também irá imprimir uma mensagem dizendo se o número é primo.

O programa

```
def prost(n):
    x='Broj je prost.'
    for i in range(2,n):
        if n%i==0:
            x='Broj nije prost.'
    return x
n=int(input('Upiši broj: '))
while n!=0:
    print(prost(n))
    print()
    n=int(input('Upiši broj: '))
```

Exemplo de teste

```
Upiši broj: 6
Broj nije prost.

Upiši broj: 7
Broj je prost.

Upiši broj: 8
Broj nije prost.

Upiši broj: 0
>>>
```

Parar o programa

EXCERCÍCIO 10

De acordo com o exemplo anterior, os alunos podem projetar, criar e testar seus próprios exemplos.

CONCLUSÃO

Alunos e professor discutem e avaliam as soluções apresentadas.

Métodos		***Formas de trabalho***
apresentação	entrevista	Trabalho individual
discussão	demonstração	Trabalho em pares
trabalhar no texto	representação	trabalho em equipa/grupo
trabalho gráfico		trabalho frontal
exercício interativo / simulação no computador		

Material:

-

Bibliografia

OBSERVAÇÕES PESSOAIS, COMENTÁRIOS E NOTAS