

FUNDAMENTOS DE PROGRAMAÇÃO

Funções

Prof. Bruno Góis Mateus



Índice

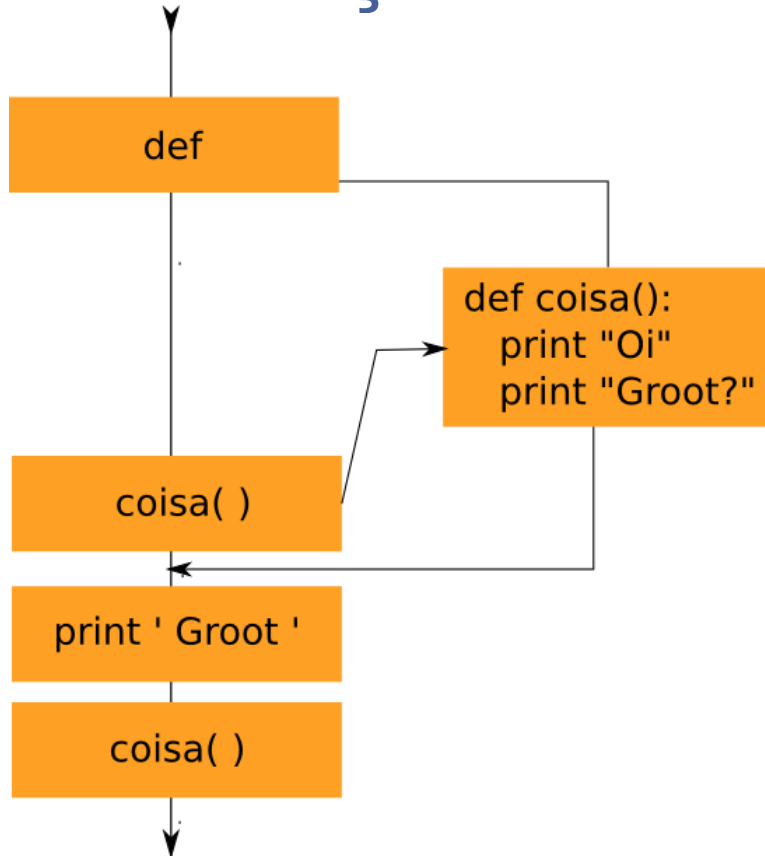
- Introdução
- Trabalhando com funções
- Criando funções
- Argumentos e Parâmetros
- Retorno
- Variáveis locais e globais
- Composição
- Exercícios

INTRODUÇÃO

Introdução

- Função
 - Sequência de instruções agrupadas
 - Quantidade ilimitada
 - Qualquer tipo de instrução
 - Utilizadas para organizar o programa em pequenos pedaços

Introdução



```
def coisa( ):
    print "Oi"
    print "Groot?"
```

```
coisa( )
print "Groot"
coisa( )
```

Oi
Groot ?
Groot
Oi
Groot ?



Introdução

- Existem dois tipos de funções em Python
 - Built-in
 - São as funções providas como parte da linguagem Python
 - `raw_input()`, `type()`, `float()`, `int()`
 - Funções
 - São aquelas que nós mesmos definimos

TRABALHANDO COM FUNÇÕES

Trabalhando com funções

- Funções são pedaços de código reusável
 - Recebem argumentos ou parâmetros
 - Realizam computações
 - Retornam o resultado das computações

Trabalhando com funções

- Existem dois tipos de funções em Python
 - Built-in
 - São as funções providas como parte da linguagem Python
 - `raw_input()`, `type()`, `float()`, `int()`
 - Funções
 - São aquelas que nós mesmos definimos


Trabalhando com funções

- Para definir uma função utilizamos a palavra reservada `def`
 - Abreviação de *definition* (definição)
- Podemos invocar/chamar uma função utilizando o seu nome, parênteses e seus argumentos em uma expressão

Trabalhando com funções

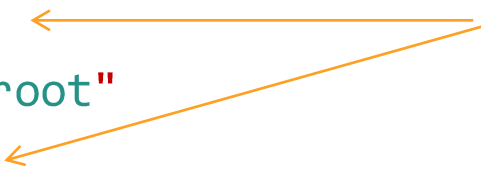
```
def coisa( ):  
    print "Oi"  
    print  
    "Groot?"
```

Definindo

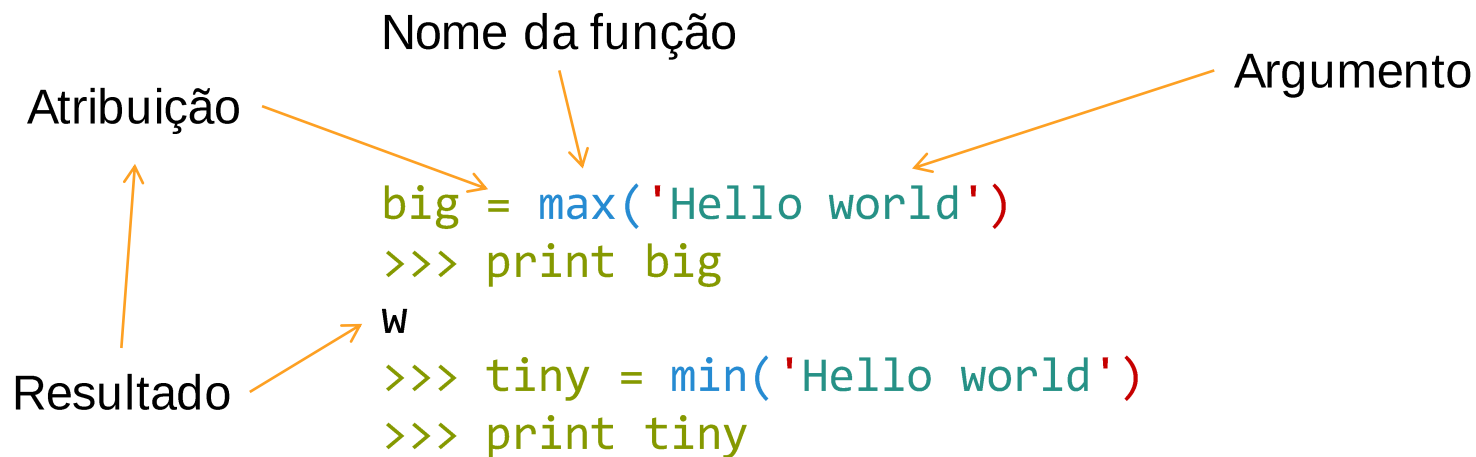


```
coisa( )  
print "Groot"  
coisa( )
```

Invocando



Trabalhando com funções



Trabalhando com funções

- Função

- Código armazenado
- Recebe uma entrada
- Retorna um saída

```
big = max('Hello world')  
>>> print big  
w
```

“Hello World”



Função
Max ()



“w”

Trabalhando com funções

- Função

- Código armazenado
- Recebe uma entrada
- Retorna um saída

```
big = max('Hello world')  
>>> print big  
w
```

“Hello World”



```
def max(inp):  
    bla  
    bla  
    bla
```



“w”

CRIANDO NOSSAS FUNÇÕES

Criando nossa funções

- Cabeçalho
 - Criamos nossa funções utilizando a palavra `def`
 - Definimos então o nome da nossa função
 - Lista de parâmetros (opcional)
 - Um ou mais parâmetros
 - Coisas que a função precisa para ser executada
 - Devem ser informados ao invocar a função

Criando nossa funções

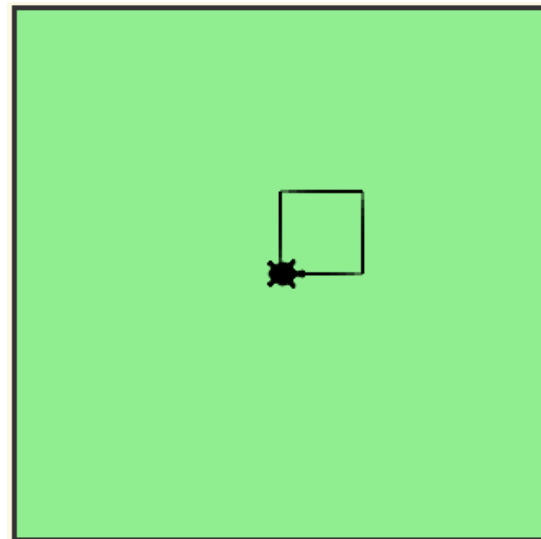
- Corpo
 - Código indentado
 - Contém as instruções da função
- Ao definir uma função, ela não será executada automaticamente

Criando nossa funções

```
import turtle

def drawSquare(t, sz):
    for i in range(4):
        t.forward(sz)
        t.left(90)

x = 5
wn = turtle.Screen()
wn.bgcolor("lightgreen")
alex = turtle.Turtle()
drawSquare(alex, 50)
x = x + 2
print x
wn.exitonclick()
```



Criando nossas funções

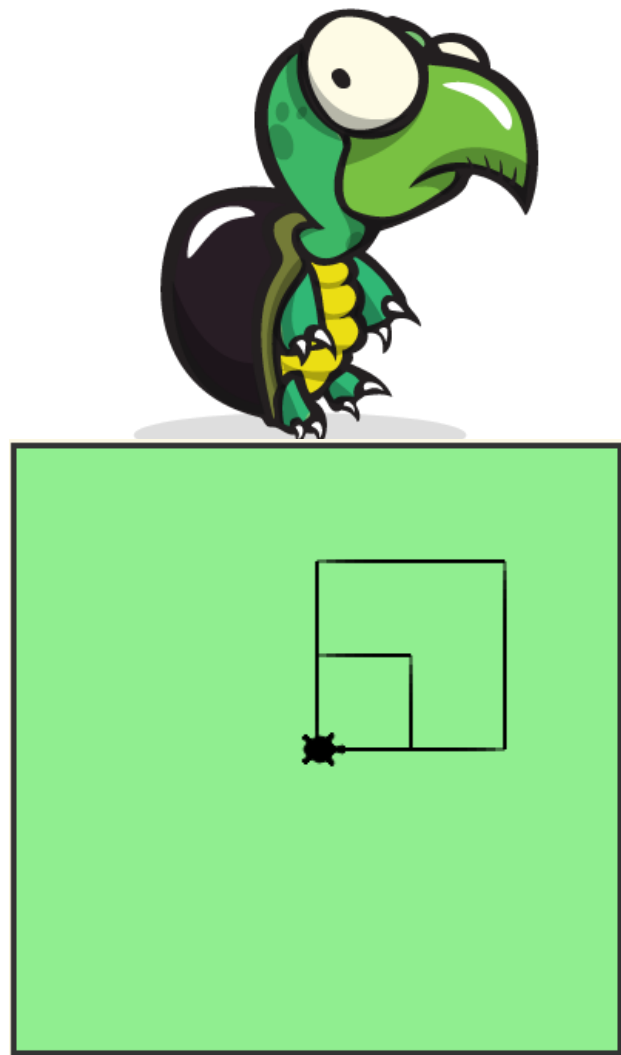
Uma vez que a função foi definido podemos utilizá-la (invocá-la, chamá-la) quantas vezes for necessário

Criando nossa funções

```
import turtle

def drawSquare(t, sz):
    for i in range(4):
        t.forward(sz)
        t.left(90)

x = 5
wn = turtle.Screen()
wn.bgcolor("lightgreen")
alex = turtle.Turtle()
drawSquare(alex, 50)
drawSquare(alex, 100)
x = x + 2
print x
wn.exitonclick()
```



ARGUMENTOS E PARÂMETROS

Argumentos

- É um valor passado como entrada de uma função
- Usamos argumentos para utilizar uma função para realizar computações “diferentes”
- São colocados entre parênteses depois do nome da função no momento da invocação

Argumento



```
big = max('Hello world')
```

Parâmetros

- São as **variáveis** que utilizamos na definição da função
- Possibilitam a manipulação dos **argumentos** dentro de uma **função**

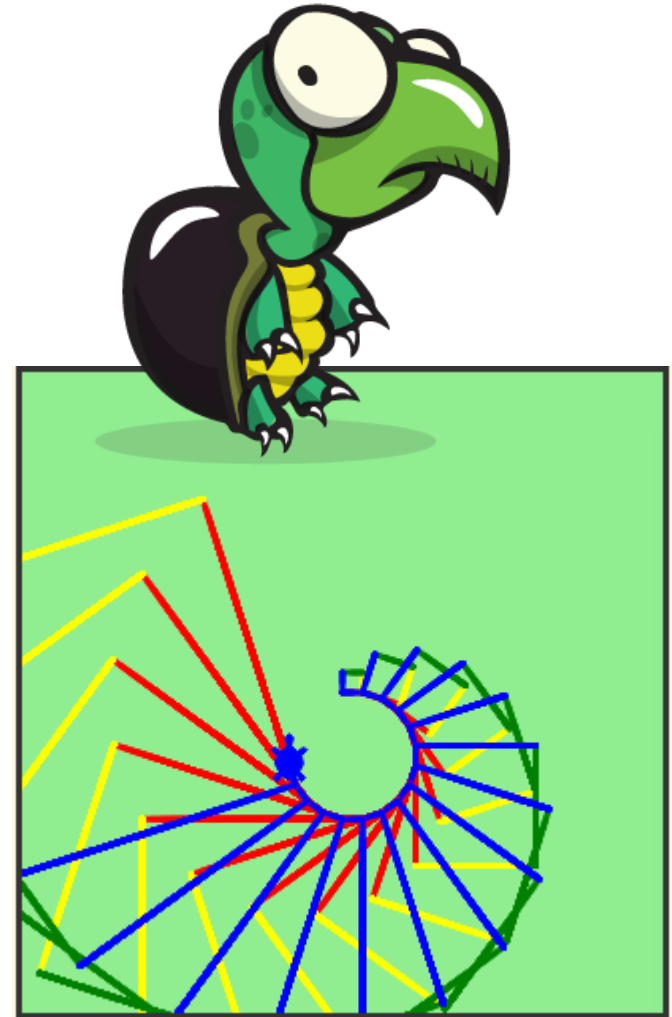
```
>> def greet(lang):  
...     if lang == 'pt':  
...         print 'Ola'  
...     elif lang == 'fr':  
...         print 'Bonjour'  
...     else:  
...         print 'Hello'  
...  
>>> greet('en')  
Hello  
>>> greet('pt')  
Ola  
>>> greet('fr')  
Bonjour  
>>>
```

Exemplo

```
def drawMulticolorSquare(t, sz):  
    for i in ['red', 'yellow', 'green', 'blue']:  
        t.color(i)  
        t.forward(sz)  
        t.left(90)
```

```
wn = turtle.Screen()  
wn.bgcolor("lightgreen")  
tess = turtle.Turtle() tess.pensize(3)  
tess.shape("turtle")
```

```
size = 10  
for i in range(15):  
    drawMulticolorSquare(tess, size)  
    size = size + 10  
    tess.forward(10)  
    tess.right(18)  
wn.exitonclick()
```



RETORNO

Retorno

- A maioria das funções precisam de argumentos
 - Algumas mais de um argumento
- Algumas funções retornam algum resultado após sua execução
 - Podem ser usadas em expressões
 - São chamadas de funções frutíferas
 - `raw_input`

```
print(max(3 * 11, 5 ** 3, 512 - 9, 1024 ** 0))
```

Retorno

```
>>> def cumprimento(lang):  
...     if lang == 'pt':  
...         return 'Ola'  
...     elif lang == 'fr':  
...         return 'Bonjour'  
...     else:  
...         return 'Hello'  
...  
>>> print greet('en'), 'Glenn'  
Hello Glenn  
>>> print greet('pt'), 'Sally'  
Ola Sally  
>>> print  
greet('fr'), 'Michael'  
Bonjour Michael  
>>>
```

Retorno

- Funções void são aquelas que não retornam algum resultado
 - Também chamadas de procedimentos em outras linguagens
 - Na prática, são funções também

Retorno

Funções com retorno



Funções sem retorno



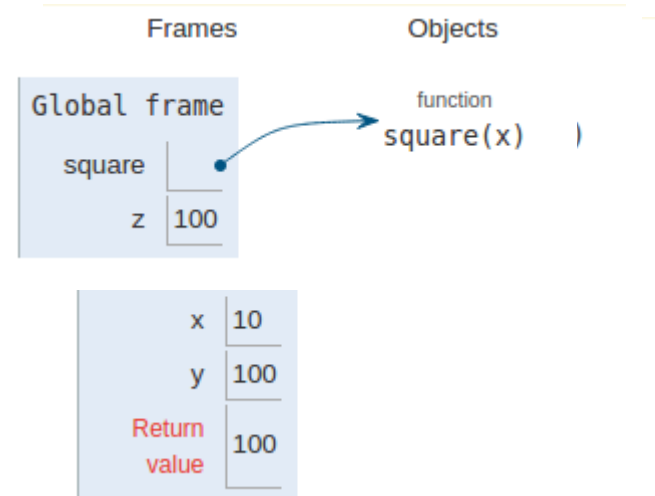
VARIÁVEIS LOCAIS E GLOBAIS

Variáveis locais

- Uma atribuição de variável dentro de uma função cria uma variável local
- Essa variável é chamada local porque ela existe apenas dentro da função
- Parâmetros são variáveis locais

Variáveis locais

```
→ def square(x):  
→     y = x * x  
→     return y  
→  
→ z = square(10)  
→ print(y)
```



Variável global

```
def gambiquadrado(x):  
    y = x ** power  
    return y  
  
power = 2  
result = badsquare(10)  
print(result)
```

Variável global

```
def powerof(x, p):  
    power = p    # Another dumb mistake  
    y = x ** power  
    return y
```

```
power = 3  
result = powerof(10, 2)  
print(result)
```

COMPOSIÇÃO

Composição

- Cada função que criamos pode:
 - Chamar outra função
 - Ser chamada por outra função
- Excelente maneira de quebrar um grande problema e em vários problemas menores
 - Pensamento muito utilizado em computação
 - Dividir para conquistar

Composição

```
def square(x):  
    y = x * x  
    return y  
  
def sum_of_squares(x,  
y, z):  
    a = square(x)  
    b = square(y)  
    c = square(z)  
  
    return a + b + c
```

```
a = -5  
b = 2  
c = 10  
result =  
sum_of_squares(a, b, c)  
print(result)
```

Calculando a área de um círculo

- Precisamos de dois pontos
 - Centro do círculo
 - Ponto no perímetro
- 1. Como calcular o raio do círculo ?
- 2. Como calcular a área do círculo ?

Generalização

- Desenhando um retângulo
- Desenhando um quadrado

Usando a função main

```
import turtle
```

```
def drawSquare(t, sz):  
    for i in range(4):  
        t.forward(sz)  
        t.left(90)
```

```
def main():  
    wn = turtle.Screen()  
    wn.bgcolor("lightgreen")  
    alex = turtle.Turtle()  
    drawSquare(alex, 50)  
    wn.exitonclick()
```

```
main()
```


EXERCÍCIOS

Exercícios

```
1 def pow(b, p):  
2     y = b ** p  
3     return y  
4  
5 def square(x):  
6     a = pow(x, 2)  
7     return a  
8  
9 n = 5  
10 result = square(n)  
11 print(result)
```

- Qual a ordem de execução das linhas?
 - a) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11
 - b) 1, 2, 3, 5, 6, 7, 9, 10, 11
 - c) 9, 10, 11, 1, 2, 3, 5, 6, 7
 - d) 9, 10, 5, 6, 7, 1, 2, 3, 11
 - e) 1, 5, 9, 10, 6, 2, 3, 7, 11

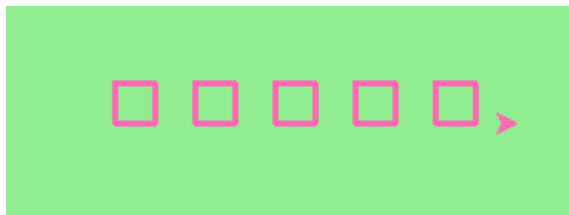
Exercícios

```
1 def pow(b, p):  
2     y = b ** p  
3     return y  
4  
5 def square(x):  
6     a = pow(x, 2)  
7     return a  
8  
9 n = 5  
10 result = square(n)  
11 print(result)
```

- Qual valor a função print irá imprimir ?
 - a) 25
 - b) 5
 - c) 125
 - d) 32

Exercícios

- Faça um programa para desenhar uma algo semelhante a imagem abaixo
 - Utilize um função para desenhar quadrados



Exercícios

- Faça um programa para desenhar uma algo semelhante a imagem abaixo
 - Utilize um função para desenhar quadrados

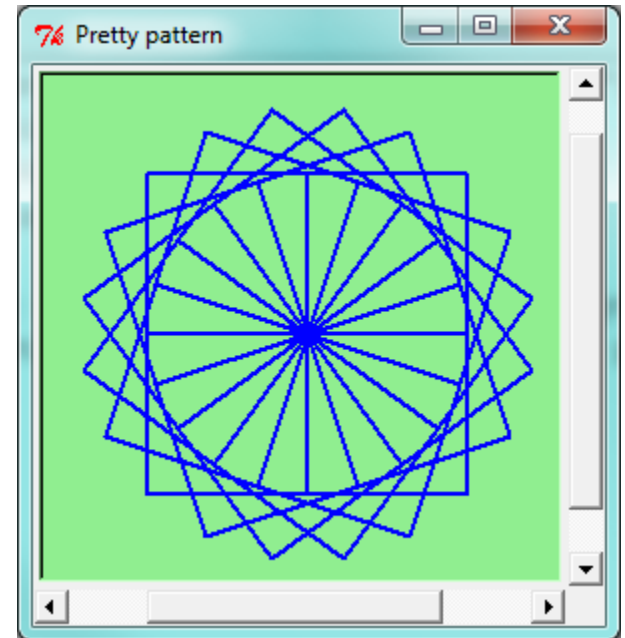


Exercícios

- Faça um programa para desenhar capaz de desenhar polígonos regulares
 - Escreva uma função capaz de desenhar tal polígono baseado no número de lados
 - A função deve receber o tamanho do lado do polígono

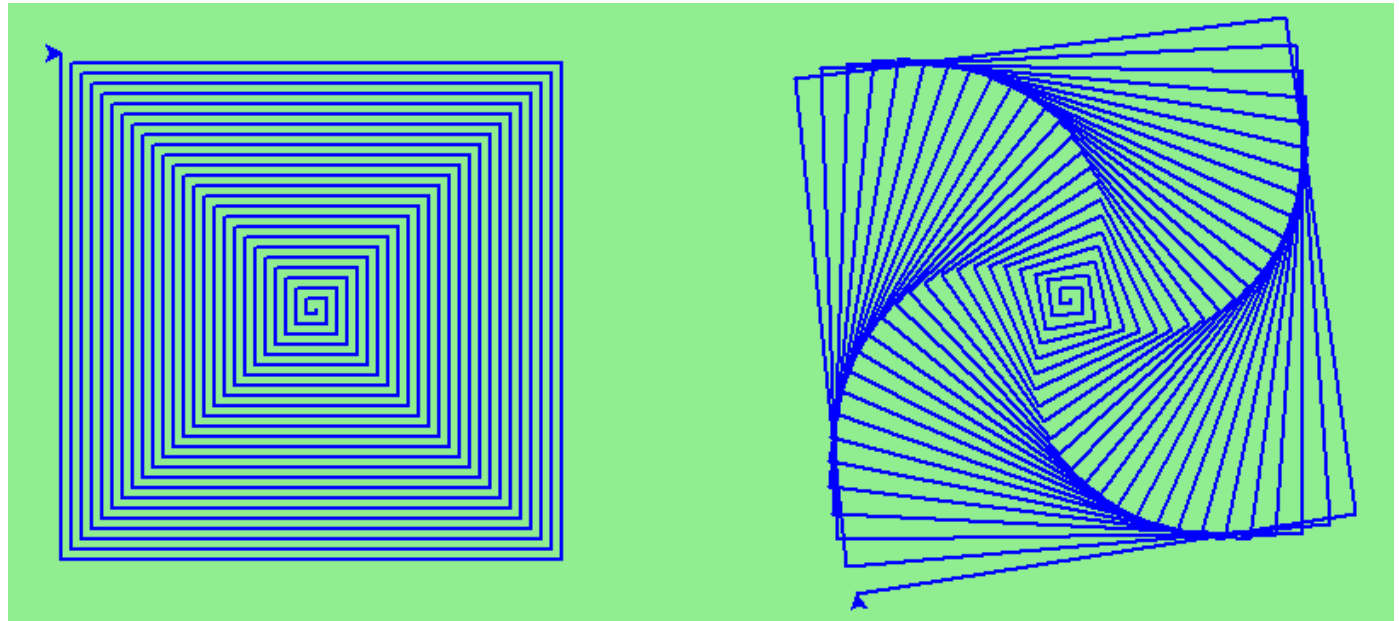
Exercícios

- Faça um programa para desenhar uma algo semelhante a imagem abaixo
 - Utilize um função para desenhar quadrados



Exercícios

- Faça um programa para desenhar uma algo semelhante a imagem abaixo
 - Dica a diferença entre essas espirais é apenas o ângulo de rotação



Exercícios

- Faça um programa para desenhar triângulos equiláteros de acordo com o tamanho do lado informado pelo usuário
 - Utilize a função da questão sobre polígonos regulares

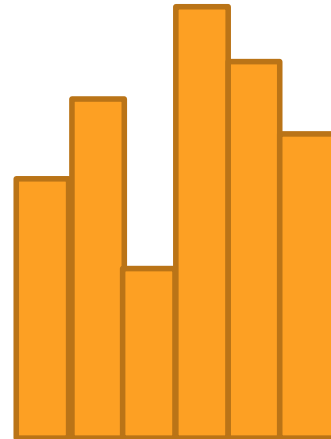
Exercícios

- Faça um programa para desenhar uma algo semelhante a imagem abaixo
 - Utilize um função para desenhar círculos



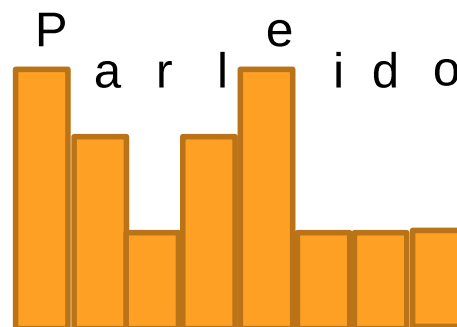
Exercícios

- Faça um programa capaz de desenhar histogramas como mostra a figura abaixo
 - A largura da barras devem ser fixas
 - A altura das barras vai depender do valor do conjunto de dados



Exercícios

- Utilize a questão anterior para desenhar um histograma relacionado a o ocorrência de cada letra em um palavra
 - Exemplo: Paralelepípedo



A seguir

- Funções