

Introdução à Linguagem Python



Code Like a Girl 2018
Joinville (SC)



Verônica Marin

26 anos

- Formação: Engenharia Elétrica - UDESC - 2016/2
- Ocupação: Model-Based Design Software Engineer - Whirlpool Corp.

- Linguagens de Programação:
 - Matlab, C/C++, C#/.NET, Python, Assembly, Java, Visual Basic
- Programa em Python desde 2016:
 - Desenvolvimento de testes automatizados para a área de Controle de Motor
 - Link entre reports e resultados de testes com plataformas colaborativas da empresa

Por que Python?

Simplicidade

Java

```
1 public class Hello
2 {
3     public static void main(String args[]) {
4         java.util.Scanner s = new java.util.Scanner(System.in);
5         System.out.print("Digite seu nome:");
6         String nome = s.nextLine();
7         System.out.println("Olá, " + nome);
8     }
9 }
```

Pascal

```
1 program HelloWorld(output);
2 var
3     nome: string;
4 begin
5     writeln('Digite seu nome:');
6     read(nome);
7     writeln('Olá, ', nome);
8 end.
```

Python

```
1 nome = input('Digite seu nome:')
2 print ('Olá, ', nome)
```

C

```
1 #include <stdio.h>
2 int main()
3 {
4     char nome[200];
5     printf("Digite seu nome:");
6     scanf("%s", nome);
7     printf("Olá, %s\n", nome);
8     return 0;
9 }
```

Multiplataforma



Jython, CPython, IronPython
(Java, C, .NET)

Robustez

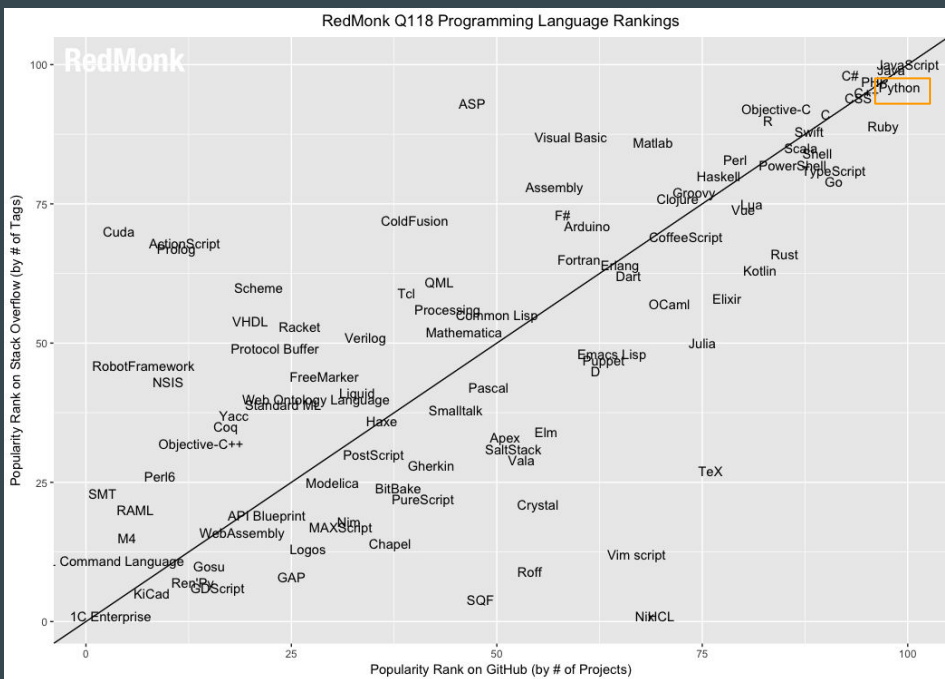
- Back-end de sistemas web, CRMs e ERPs
- Pesadas simulações de engenharia
- Processamento pesado de efeitos especiais de filmes
- Soluções de análise de dados (data analytics)
- Aprendizado de máquina (machine learning – ML)

Comunidade Python























- Python Software Foundation
- Associação Python Brasil
- PyLadies
- Django Girls



Ranking de Linguagens de Programação



Fonte: <http://redmonk.com/>
Janeiro 2018

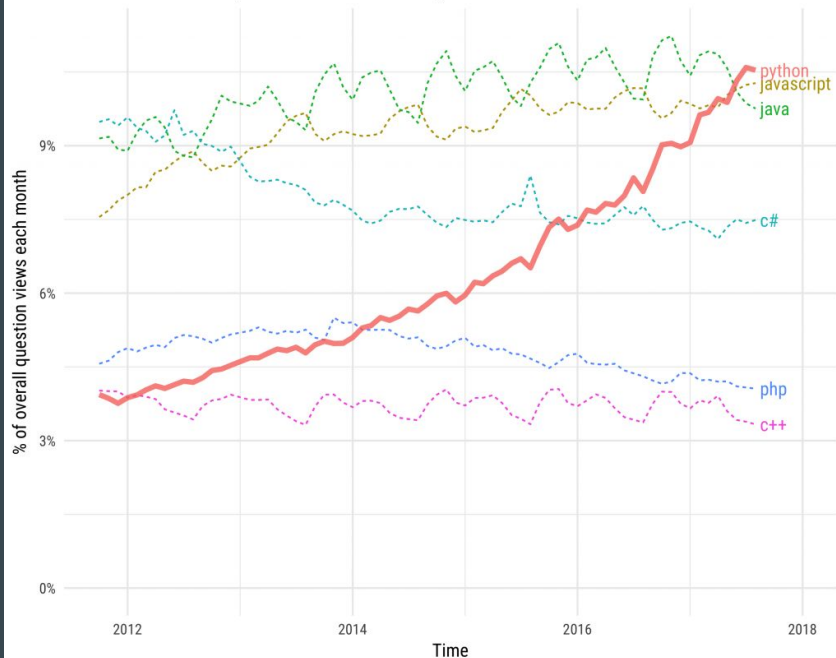
Language Rank	Types	Spectrum Ranking
1. Python	 	100.0
2. C	  	99.7
3. Java	  	99.5
4. C++	  	97.1
5. C#	  	87.7
6. R		87.7
7. JavaScript	 	85.6
8. PHP		81.2
9. Go	 	75.1
10. Swift	 	73.7

Fonte: <https://spectrum.ieee.org/>
2017 Top Programming Languages

Crescimento e Comunidade Global

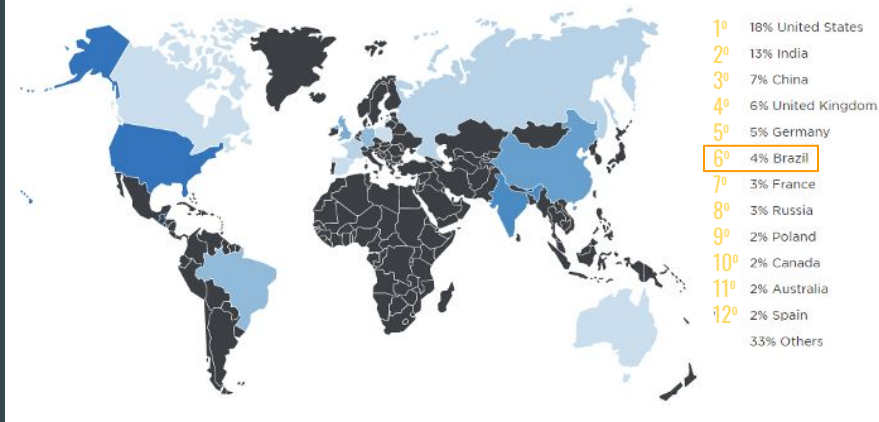
Growth of major programming languages

Based on Stack Overflow question views in World Bank high-income countries



Fonte: <https://stackoverflow.blog/2017/09/06/incredible-growth-python/>
The Incredible Growth of Python - Stack Overflow

Global Community



Fonte: <http://ib.gg/pythondevsurvey2017>
Python Developers Survey 2017

Acesse a apresentação:

<https://tinyurl.com/code-like-a-girl-2018>

Por onde começo?

Para começar a programar em Python você vai precisar de:

- Python (<https://www.python.org/downloads/>)
- [Editor de Código](#) ou IDE (Ambiente de Desenvolvimento Integrado)

Um código em Python estará em um arquivo “.py”

Para executar um programa em Python utilizaremos a [Linha de Comando](#):

```
$ python meuprograma.py
```

Módulos

In [6]:

```
import math
```

In [7]:

```
import math

x = math.cos(2 * math.pi)

print(x)

1.0
```

In [8]:

```
from math import *

x = cos(2 * pi)

print(x)

1.0
```

In [9]:

```
from math import cos, pi

x = cos(2 * pi)

print(x)

1.0
```

Listar todas as funções de um módulo:

```
print(dir(math))
```

Descrição e documentação da função:

```
help(math.log)
```

```
log(...)
log(x[, base])
```

Return the logarithm of x to the given base.

If the base not specified, returns the natural logarithm (base e) of x.

Python 2 Standard Library (<https://docs.python.org/2/library/>)

Python 3 Standard Library (<https://docs.python.org/3/library/>)

Exercício - Módulos

1) Importando o módulo “math” calcule:

- a) $x = \cos(\pi/2) + \sin(\pi/2)$ = 1.0
- b) $x = \sqrt{3}$ = 1.73205080757
- c) $x = \log_2 10$ = 3.3219280948873626

2) Importando o módulo “calendar” descubra:

- a) 2018 é um ano bissexto (leap year)? R: False (Não)
- b) Dia 22 de Maio de 1992 foi que dia da semana? R: 4 (Sexta)
- c) O mês de Julho de 2000 começou em que dia da semana? R: 5 (Sábado)

Variáveis e Tipos

- O nome de uma variável pode conter caracteres alfanumericos (a-z)(A-Z)(0-9) e alguns caracteres especiais (_).
- O nome deve começar com uma letra ou com underscore (_).

```
# variable assignments
x = 1.0
my_variable = 12.2
```

Python automaticamente define o tipo da variável e o seu local na memória.

```
i = 42          # data type is implicitly set to integer
i = 42 + 0.11   # data type is changed to float
i = "forty"     # and now it will be a string
```

Para acessar o tipo associado a variável devemos utilizar `type(x)`

and, as, assert, break, class, continue, def, del, elif, else, except, exec, finally, for, from, global, if, import, in, is, lambda, not, or, pass, print, raise, return, try, while, with, yield

Python Keywords - Não podem ser nomes de variáveis

- Tipos Fundamentais:

```
In [19]: # integers
         x = 1
         type(x)
Out[19]: int

In [20]: # float
         x = 1.0
         type(x)
Out[20]: float

In [21]: # boolean
         b1 = True
         b2 = False
         type(b1)
Out[21]: bool

In [22]: # complex numbers: note the use of `j` to specify the imaginary part
         x = 1.0 - 1.0j
         type(x)
Out[22]: complex

In [23]: print(x)
         (1-1j)

In [24]: print(x.real, x.imag)
         (1.0, -1.0)
```

Variáveis e Tipos

- Tipos Fundamentais:

```
In [19]: # integers
x = 1
type(x)
```

```
Out[19]: int
```

```
In [20]: # float
x = 1.0
type(x)
```

```
Out[20]: float
```

```
In [21]: # boolean
b1 = True
b2 = False

type(b1)
```

```
Out[21]: bool
```

```
In [22]: # complex numbers: note the use of `j` to specify the imaginary part
x = 1.0 - 1.0j
type(x)
```

```
Out[22]: complex
```

```
In [23]: print(x)

(1-1j)
```

```
In [24]: print(x.real, x.imag)

(1.0, -1.0)
```

- Casting:

```
In [29]: x = 1.5

print(x, type(x))

(1.5, <type 'float'>)
```

```
In [30]: x = int(x)

print(x, type(x))

(1, <type 'int'>)
```

```
In [31]: z = complex(x)

print(z, type(z))

((1+0j), <type 'complex'>)
```

```
In [32]: x = float(z)
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-32-e719cc7b3e96> in <module>()
----> 1 x = float(z)

TypeError: can't convert complex to float
```

Complex variables cannot be cast to floats or integers. We need to use `z.real` or `z.imag` to extract the part of the complex number we want:

```
In [33]: y = bool(z.real)

print(z.real, " -> ", y, type(y))

y = bool(z.imag)

print(z.imag, " -> ", y, type(y))

(1.0, ' -> ', True, <type 'bool'>)
(0.0, ' -> ', False, <type 'bool'>)
```

Exercício - Variáveis e Tipos

1) Defina as variáveis $x = 1$, $y = 2.3$ e $z = x+y$. Qual o tipo de x , y e z ?

R: $x = \text{int}$, $y = \text{float}$, $z = \text{float}$

2) Defina as variáveis $x = 1$, $y = 4$ e $z = x/y$. Qual o valor de z ?

R: 0

3) Defina as variáveis $x = 1.0$, $y = 4$ e $z = x/y$. Qual o valor de z ?

R: 0.25

4) Existe outra forma de obter 0.25 sem definir x como 1.0, você sabe qual é? Se sim, implemente-a.

Operadores e Comparações

- Operadores aritméticos:

+	Soma
-	Subtração
*	Multiplicação
/	Divisão
**	Potência (atenção, não é “^”)
//	Divisão de Inteiros
%	Resto da Divisão

- Comparações

== or is	Igual
!= or <>	Diferente
>	Maior
<	Menor
>=	Maior ou Igual
<=	Menor ou Igual

- Operadores booleanos:

In [38]:

```
True and False
```

Out[38]:

False

In [39]:

```
not False
```

Out[39]:

True

In [40]:

```
True or False
```

Out[40]:

True

Exercício - Operadores e Comparações

1) Defina a variável $x = 7$

- | | | |
|----|-----------------------------|---------|
| a) | $x * 2$ é maior que 10? | R: True |
| b) | $x / 3$ é menor que 5? | R: True |
| c) | X ao quadrado é igual a 49? | R: True |

2) Defina a variável $y = 3$

- | | | |
|----|----------------------------------------------|----------|
| a) | y é menor que 10 e x é maior que 10? | R: False |
| b) | y é maior ou igual a 3 ou x é igual a 8? | R: True |
| c) | y não é igual a 4? | R: True |

Tipos Complexos - Strings

- Usadas para guardar mensagens de texto.

```
In [46]: s = "Hello world"
         type(s)
```

```
Out[46]: str
```

```
In [47]: # length of the string: the number of characters
         len(s)
```

```
Out[47]: 11
```

```
In [48]: # replace a substring in a string with something else
         s2 = s.replace("world", "test")
         print(s2)
```

```
Hello test
```

Funções Úteis:

- s.upper() - Todas as letras em caixa alta
- s.lower() - Todas as letras em caixa baixa
- s.find("expressão")
- s.split("separador")

- É possível acessar apenas um pedaço da string usando [início : fim : step], o valor padrão de step é 1

```
In [49]:
```

```
s[0]
```

```
Out[49]:
```

```
'H'
```

```
In [50]:
```

```
s[0:5]
```

```
Out[50]:
```

```
'Hello'
```

```
In [51]:
```

```
s[4:5]
```

```
Out[51]:
```

```
'o'
```

```
In [52]:
```

```
s[:5]
```

```
Out[52]:
```

```
'Hello'
```

```
In [53]:
```

```
s[6:]
```

```
Out[53]:
```

```
'world'
```

```
In [54]:
```

```
s[:]
```

```
Out[54]:
```

```
'Hello world'
```

```
In [56]:
```

```
s[::-2]
```

```
Out[56]:
```

```
'Hlowrd'
```

Tipos Complexos - Lists

- Usadas para uma lista de itens.

```
In [63]:  
l = [1,2,3,4]  
  
print(type(l))  
print(l)  
  
<type 'list'>  
[1, 2, 3, 4]
```

Funções Úteis:

- | | |
|----------------------------|---------------------------------|
| - range(inicio, fim, step) | - Retorna uma lista |
| - l.sort() | - Organiza por ordem alfabética |
| - l.append(item) | - Adiciona item ao fim da lista |
| - l.insert(index,item) | - Adiciona item ao index |
| - l.remove(item) | - Remove item da lista |
| - l.reverse() | - Inverte a ordem da lista |

- Assim como strings, é possível acessar apenas um pedaço da lista usando [início : fim : step]

```
In [64]:  
print(l)  
print(l[1:3])  
print(l[:2])  
  
[1, 2, 3, 4]  
[2, 3]  
[1, 3]
```

- Os itens não precisam ser do mesmo tipo

```
In [66]:  
l = [1, 'a', 1.0, 1-1j]  
print(l)  
  
[1, 'a', 1.0, (1-1j)]
```

Tipos Complexos - Tuples

- Usadas para uma lista IMUTÁVEL de itens.

```
In [80]: point = (10, 20)
print(point, type(point))
((10, 20), <type 'tuple'>)
```

```
In [81]: point = 10, 20
print(point, type(point))
((10, 20), <type 'tuple'>)
```

```
In [82]: x, y = point
print("x =", x)
print("y =", y)
('x =', 10)
('y =', 20)
```

Python Expression	Results	Description
len((1, 2, 3))	3	Length
(1, 2, 3) + (4, 5, 6)	(1, 2, 3, 4, 5, 6)	Concatenation
('Hi!') * 4	('Hi!', 'Hi!', 'Hi!', 'Hi!')	Repetition
3 in (1, 2, 3)	True	Membership
for x in (1, 2, 3): print x,	1 2 3	Iteration

Tipos Complexos - Dictionaries

- Usadas para uma lista de itens e chaves.

Funções Úteis:

- `dict.copy()`
- `dict.has_key()` - Verifica se chave existe
- `dict.values()` - Retorna uma lista de itens

```
In [84]: params = {"parameter1" : 1.0,
                  "parameter2" : 2.0,
                  "parameter3" : 3.0,}

print(type(params))
print(params)

<type 'dict'>
{'parameter1': 1.0, 'parameter3': 3.0, 'parameter2': 2.0}
```

```
In [85]: print("parameter1 = " + str(params["parameter1"]))
print("parameter2 = " + str(params["parameter2"]))
print("parameter3 = " + str(params["parameter3"]))

parameter1 = 1.0
parameter2 = 2.0
parameter3 = 3.0
```

```
In [86]: params["parameter1"] = "A"
params["parameter2"] = "B"

# add a new entry
params["parameter4"] = "D"

print("parameter1 = " + str(params["parameter1"]))
print("parameter2 = " + str(params["parameter2"]))
print("parameter3 = " + str(params["parameter3"]))
print("parameter4 = " + str(params["parameter4"]))

parameter1 = A
parameter2 = B
parameter3 = 3.0
parameter4 = D
```

Fluxo Condicional - if, elif e else

```
In [87]: statement1 = False
         statement2 = False

         if statement1:
             print("statement1 is True")
         elif statement2:
             print("statement2 is True")
         else:
             print("statement1 and statement2 are False")

         statement1 and statement2 are False
```

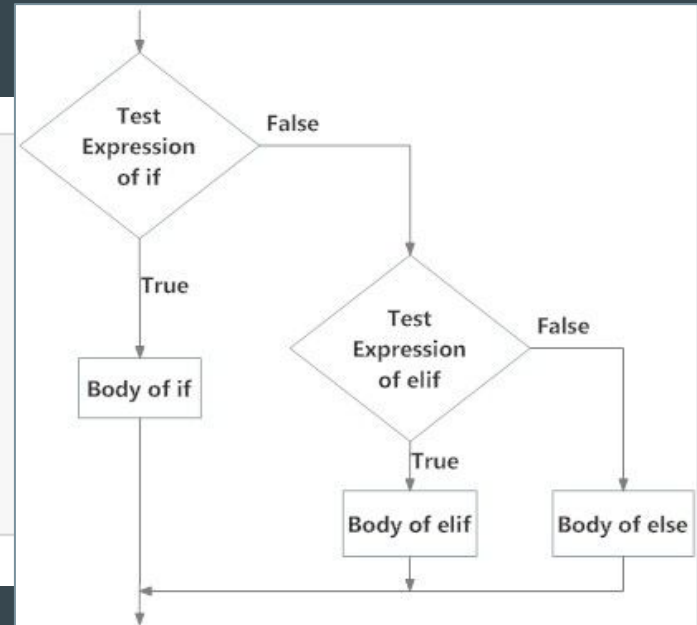


Fig: Operation of if...elif...else statement

Loops - for

```
In [93]: for x in range(4): # by default range start at 0
         print(x)
```

```
0
1
2
3
```

```
In [94]: for x in range(-3,3):
         print(x)
```

```
-3
-2
-1
0
1
2
```

```
In [95]: for word in ["scientific", "computing", "with", "python"]:
         print(word)
```

```
scientific
computing
with
python
```

```
In [96]: for key, value in params.items():
         print(key + " = " + str(value))
```

```
parameter4 = D
parameter1 = A
parameter3 = 3.0
parameter2 = B
```

- For loop na criação de listas:

```
In [98]: l1 = [x**2 for x in range(0,5)]
         print(l1)
```

```
[0, 1, 4, 9, 16]
```

Loops - while

In [99]:

```
i = 0  
  
while i < 5:  
    print(i)  
  
    i = i + 1  
  
print("done")
```

```
0  
1  
2  
3  
4  
done
```

Funções

```
In [100]: def func0():  
          print("test")
```

```
In [101]: func0()  
  
test
```

```
In [105]: def square(x):  
          """  
          Return the square of x.  
          """  
          return x ** 2
```

```
In [106]: square(4)
```

```
Out[106]: 16
```

```
In [107]: def powers(x):  
          """  
          Return a few powers of x.  
          """  
          return x ** 2, x ** 3, x ** 4
```

```
In [108]: powers(3)
```

```
Out[108]: (9, 27, 81)
```

```
In [109]: x2, x3, x4 = powers(3)  
  
print(x3)
```

```
27
```

- Argumento padrão:

```
In [110]: def myfunc(x, p=2, debug=False):  
          if debug:  
              print("evaluating myfunc for x = " + str(x) + " using exponent p = " + str(p))  
          return x**p
```

Funções sem nome (lambda)

```
In [114]: f1 = lambda x: x**2
```

```
# is equivalent to
```

```
def f2(x):  
    return x**2
```

```
In [115]: f1(2), f2(2)
```

```
Out[115]: (4, 4)
```

```
In [116]: # map is a built-in python function
```

```
map(lambda x: x**2, range(-3,4))
```

```
Out[116]: [9, 4, 1, 0, 1, 4, 9]
```

Classes

```
In [118]: class Point:
    """
    Simple class for representing a point in a Cartesian coordinate system.
    """

    def __init__(self, x, y):
        """
        Create a new Point at x, y.
        """
        self.x = x
        self.y = y

    def translate(self, dx, dy):
        """
        Translate the point by dx and dy in the x and y direction.
        """
        self.x += dx
        self.y += dy

    def __str__(self):
        return("Point at [%f, %f]" % (self.x, self.y))
```

```
In [120]: p2 = Point(1, 1)
p1.translate(0.25, 1.5)

print(p1)
print(p2)

Point at [0.250000, 1.500000]
Point at [1.000000, 1.000000]
```

Módulos

```
In [121]: %%file mymodule.py
          """
          Example of a python module. Contains a variable called my_variable,
          a function called my_function, and a class called MyClass.
          """

          my_variable = 0

          def my_function():
              """
              Example function
              """
              return my_variable

          class MyClass:
              """
              Example class.
              """

              def __init__(self):
                  self.variable = my_variable

              def set_variable(self, new_value):
                  """
                  Set self.variable to a new value
                  """
                  self.variable = new_value

              def get_variable(self):
                  return self.variable
```

```
In [122]: import mymodule
```


Exceções

```
In [128]: raise Exception("description of the error")
```

```
-----  
Exception                                 Traceback (most recent call last)  
<ipython-input-128-8f47ba831d5a> in <module>()  
----> 1 raise Exception("description of the error")  
  
Exception: description of the error
```

```
def my_function(arguments):  
  
    if not verify(arguments):  
        raise Exception("Invalid arguments")  
  
    # rest of the code goes here
```

```
In [129]: try:  
           print("test")  
           # generate an error: the variable test is not defined  
           print(test)  
       except:  
           print("Caught an exception")  
  
test  
Caught an exception
```

```
In [130]: try:  
           print("test")  
           # generate an error: the variable test is not defined  
           print(test)  
       except Exception as e:  
           print("Caught an exception:" + str(e))  
  
test  
Caught an exception:name 'test' is not defined
```

Exercícios Finais - Funções e Loops

- 1) Crie uma função que multiplica todos os itens de uma lista.
 - Entrada: (8, 2, 3, -1, 7)
 - Saída: -336

- 2) Crie uma função que retorna o fatorial de um número com uma entrada positiva. Levantar uma exceção em caso de entrada negativa.
 - Entrada: 8
 - Saída: 40320

- 3) Crie uma função com quatro entradas: divisor, multiplicador, limite mínimo e limite máximo. Retornar todos os números entre os limites que seja divisível pelo divisor e múltiplo do multiplicador.
 - Entrada: (7,5,1500,1700)
 - Saída: (1505,1540,1575,1610,1645,1680)

Exercícios Finais - Classes e Módulos

- 1) Crie uma classe chamada retângulo, com um comprimento, uma largura e uma função que retorna a área do retângulo.

$$\text{Área do Retângulo} = \text{Comprimento} \times \text{Largura}$$

- 2) Crie uma classe chamada círculo, com um raio, uma função que retorna a área e uma função que retorna o comprimento do círculo.

$$\text{Área do Círculo} = \pi \times \text{Raio}^2$$

$$\text{Comprimento do Círculo} = 2 \times \pi \times \text{Raio}$$

- 3) Crie um módulo com as classes retângulo e círculo, chame-o de formas geométricas e importe em outro arquivo. Encontre a diferença entre a área de um círculo com raio 5cm e a área de um quadrado de lado 5cm.
 $R = 53,53 \text{ cm}^2$

Mais informações:

- [Introduction to Python Programming](#)
- <http://www.python.org>
- <http://www.python.org/dev/peps/pep-0008>
- Página oficial da linguagem de programação Python
- Guia de estilo de código para programação Python - **Recomendado.**

Exercícios e Tutoriais:

- <https://www.w3resource.com/python-exercises/>
- <https://www.codecademy.com/learn/learn-python>
- <https://py.checkio.org/>

Cola

Exercício - Módulos

1) Importando o módulo “math” calcule:

- a) $\cos(\pi/2) + \sin(\pi/2)$ = `math.cos(math.pi/2) + math.sin(math.pi/2)`
- b) $\sqrt{3}$ = `math.sqrt(3)`
- c) $\log_2 10$ = `math.log(10,2)` = `log(x,base)`

2) Importando o módulo “calendar” descubra:

- a) 2018 é um ano bissexto (leap year)? R: `calendar.isleap(2018)`
- b) Dia 22 de Maio de 1992 foi que dia da semana? R: `calendar.weekday(1992,5,22) = weekday(ano,mês,dia)`
- c) O mês de Julho de 2000 começou em que dia da semana? R: `calendar.monthrange(2000,7) = monthrange(ano,mês)`

Exercício - Variáveis e Tipos

4) Existe outra forma de obter 0.25 sem definir x como 1.0, você sabe qual é? Se sim, implemente-a.

R: `x = 1, y = 4,`
`z = float(x)/y`

Exercício - Operadores e Comparações

1) Defina a variável $x = 7$

- a) $x * 2$ é maior que 10?
- b) $x / 3$ é menor que 5?
- c) x ao quadrado é igual a 49?

R: $x * 2 > 10$

R: $x / 3 < 5$

R: $x ** 2 == 49$

2) Defina a variável $y = 3$

- a) y é menor que 10 e x é maior que 10?
- b) y é maior ou igual a 3 ou x é igual a 8?
- c) y não é igual a 4?

R: $y < 10$ and $x > 10$

R: $y <= 3$ or $x == 8$

R: not $y == 4$ / $y != 4$ / $y <> 4$