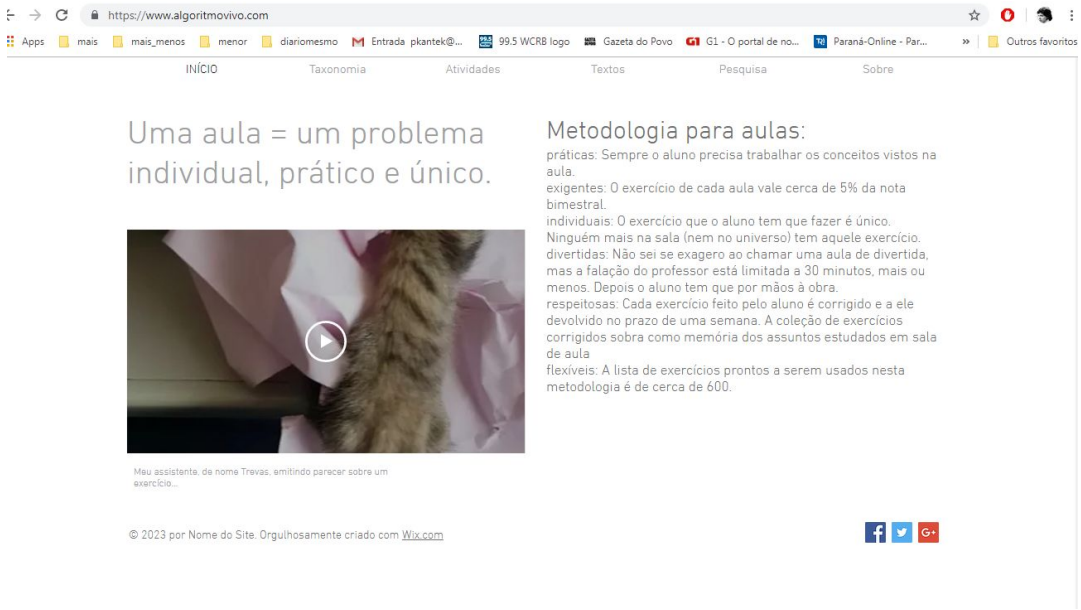
Pedro Kantek

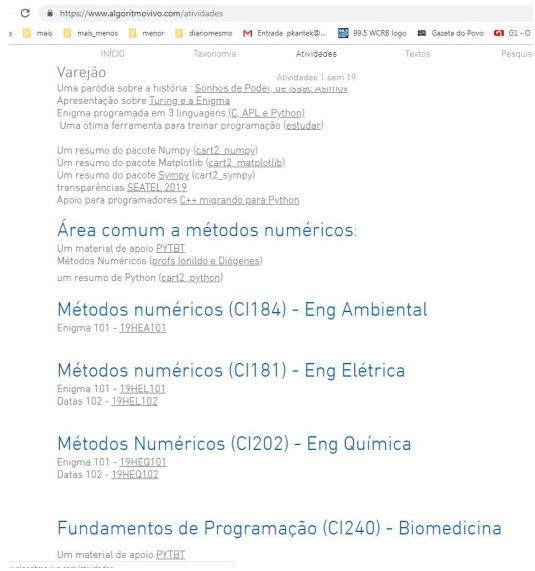
PYTHON E SEUS 170.000 PACOTES

Curitiba

2019

UFPR


https://www.algoritmovivo.com
Apps mais mais_menos menor diariomesmo Entrada pkantek@... 99.5 WCRB logo Gazeta do Povo G1 - O portal de no... Paraná-Online - Par... Outros favoritos
INÍCIO Taxonomia Atividades Textos Pesquisa Sobre
Uma aula = um problema individual, prático e único.
Metodologia para aulas:
práticas: Sempre o aluno precisa trabalhar os conceitos vistos na aula.
exigentes: O exercício de cada aula vale cerca de 5% da nota bimestral.
individuais: O exercício que o aluno tem que fazer é único. Ninguém mais na sala (nem no universo) tem aquele exercício.
divertidas: Não sei se exagero ao chamar uma aula de divertida, mas a falação do professor está limitada a 30 minutos, mais ou menos. Depois o aluno tem que por mãos à obra.
respeitosas: Cada exercício feito pelo aluno é corrigido e a ele devolvido no prazo de uma semana. A coleção de exercícios corrigidos sobra como memória dos assuntos estudados em sala de aula
flexíveis: A lista de exercícios prontos a serem usados nesta metodologia é de cerca de 600.
Meu assistente, de nome Travas, emitindo parecer sobre um exercício...
© 2023 por Nome do Site. Orgulhosamente criado com Wix.com


https://www.algoritmovivo.com/atividades
INÍCIO Taxonomia Atividades Textos Pesquis
Varejão
Atividades 1.sem 19
Uma paródia sobre a história : Sonhos de Poder, de Ioaqu Asimov
Apresentação sobre Turing e a Enigma
Enigma programada em 3 linguagens (C, APL e Python)
Uma ótima ferramenta para treinar programação (estudar)
Um resumo do pacote Numpy (cart2_numpy)
Um resumo do pacote Matplotlib (cart2_matplotlib)
Um resumo do pacote Sympy (cart2_sympy)
transparências SEATEL 2019
Apoio para programadores C++ migrando para Python
Área comum a métodos numéricos:
Um material de apoio PYTBT
Métodos Numéricos (profs Ionildo e Diógenes)
um resumo de Python (cart2_python)
Métodos numéricos (CI184) - Eng Ambiental
Enigma 101 - 19HEA101
Métodos numéricos (CI181) - Eng Elétrica
Enigma 101 - 19HEL101
Datas 102 - 19HEL102
Métodos Numéricos (CI202) - Eng Química
Enigma 101 - 19HEQ101
Datas 102 - 19HEQ102
Fundamentos de Programação (CI240) - Biomedicina
Um material de apoio PYTBT

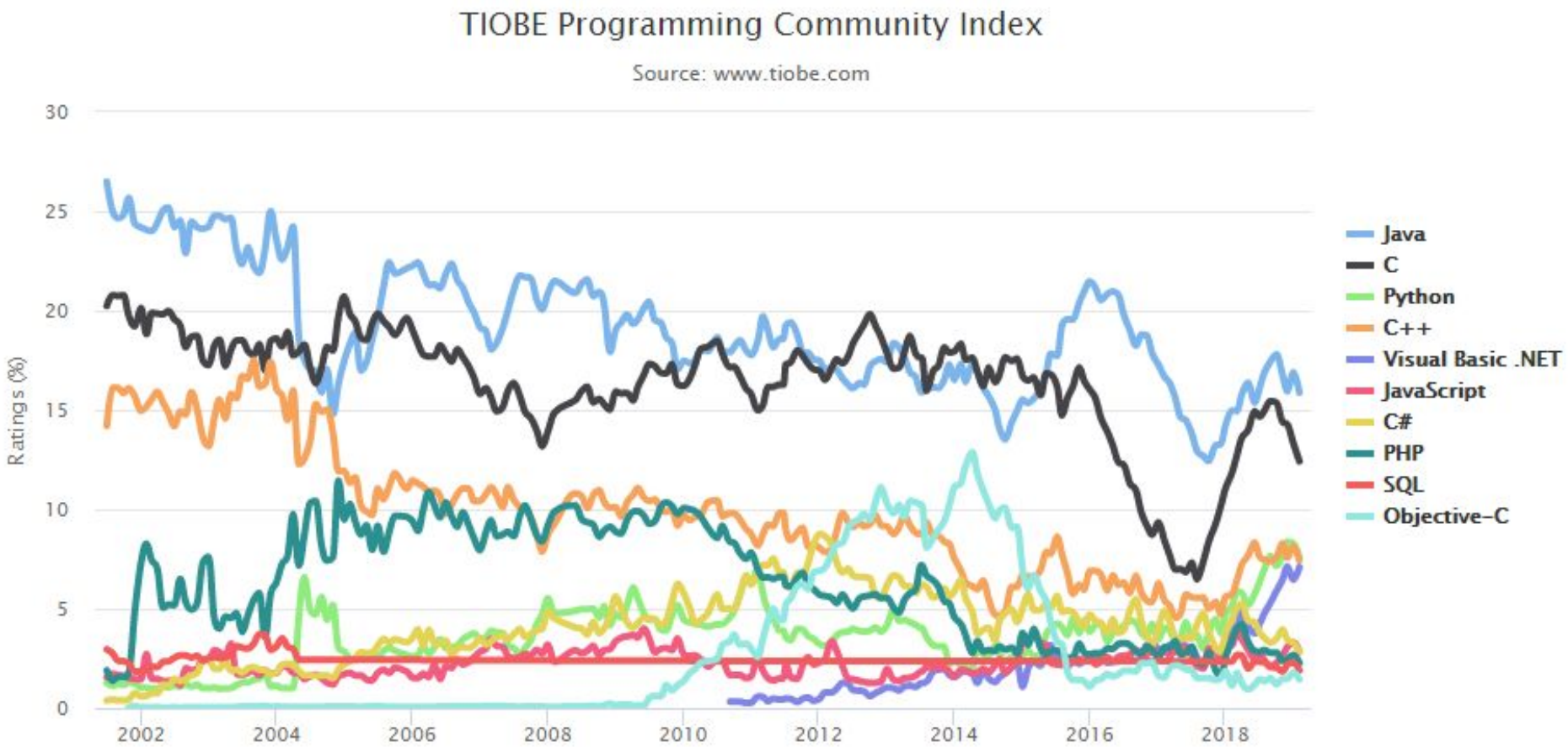
Por que Python é o cara ?

- Nasceu na hora certa
  - 2000
  - abundância de recursos
  - múltiplos modelos (copia do que é bom)
  - Infraestrutura de desenvolvimento global
- muito fácil de aprender
- resiliente

Python

- Guido Van Rossum
- Python vem de Monthy Python

TIOBE


TIOBE Programming Community Index
Source: www.tiobe.com
Ratings (%)
30
25
20
15
10
5
0
2002
2004
2006
2008
2010
2012
2014
2016
2018
Java
C
Python
C++
Visual Basic .NET
JavaScript
C#
PHP
SQL
Objective-C

Porque Python

- Moderna, veja-se por exemplo
  - big nums; / e //; matemática complexa; Unicode e UTF
  - tipagem só a nosso favor
- Chegou na hora exata para desenvolvimento global. (Github c/ 65Mproj)
- Multiplataforma: Unix, Windows, Apple, celular android, raspberry e outros
- Freeware
- Mainstream educativo
  - escrita limpa: não há ruído
  - 27 das 40 maiores universidades nos EUA usam-no
- Pacotes, pacotes, pacotes, também todos free
- Inteligência Artificial (DistBelief $\rightarrow$ Tensorflow)

Comparando

Em C++

```
#include<iostream>
#include<cmath>
using namespace std;
float pita(float a, float b){
   return sqrt(pow(a,2)+pow(b,2));
   }
int main(){
   float ca,cb,hp;
   cout<<"Informe o cateto A: "<< endl;
   cin>>ca;
   cout<<"Informe o cateto B: "<< endl;
   cin>>cb;
   hp = pita(ca,cb);
   cout<<"A hipotenusa eh: "<<hp;
}
```

Em Python

```
def pita(a,b):
  return (a**2+b**2)**0.5
ca=float(input("Informe o cateto A: "))
cb=float(input("Informe o cateto B: "))
hp=pita(ca,cb)
print("A hipotenusa é: ",hp)
```

Versões

Python 3.7.3	liberado em 25 março 2019
Python 3.7	liberado em 27 junho 2018
Python 3.6.5	liberado em 28 Março 2018
Python 3.6.4	19 Dezembro 2017
Python 3.6.0	23 Dezembro 2016
Python 3.5.1	07 Dezembro 2015
Python 3.3.7	19 Setembro 2017
Python 3.2.6	11 Outubro 2014
Python 3.0	3 Dezembro 2008
Python 2.7.8	1 Julho 2014
Python 2.6.2	14 Abril 2009
Python 2.1.3	8 Abril 2002
Python 1.4	25 Outubro 1996

algumas opções de instalação

- www.python.org: o mais indicado
  - Windows
    - 32 bits
    - 64 bits
  - Linux
  - Mac OSX
- winpython.sourceforge.net: portável, pacotes incluídos
- ipython (UFPR: elétrica): ambiente interativo
- www.pythonanywhere.com: na web
- qpython3 for android
- www.w3schools.com/python: auto-estudo com experimentação
- ...

Instalação

www.python.org
processo de instalação bem simples:

- o interpretador
- um editor de programas
- o ambiente de instalação de pacotes (PIP)

Função e script

é uma distinção semântica: uma função tem nome e pode ser referenciada depois. Precisa ser salva como um arquivo
Um script é salvo como arquivo e daí executado, como se tivesse sido recém digitado (exceto as impressões)
O caminho das pedras:

1. Chamar o python
2. File-new
3. digitar o script ou a função
4. F5
5. dar-lhe o nome de arquivo a salvar
6. observar o resultado

Tipos básicos

- inteiro
- float
- complex
- bool (True e False)
- str - entre aspas (simples ou duplas)

O tipo não é obrigatório, O Python tem tipagem forte e dinâmica e portanto pode escolher (e geralmente escolhe bem)

Nomes

"coisas" precisam ser nomeadas: funções, variáveis, módulos, classes, etc etc.

Regra `a..zA..Z seguido de a..zA..Z_0..9`

- diacríticos podem ser usados (mas não devem)
- palavras reservadas proibidas
- minúsculas $\neq$ MAIÚSCULAS

Regra: nomes que façam sentido para você

Assinalamento de variáveis

Como em C++, é o sinal =

Em a=b, a expressão *b* é avaliada e seu valor associado à variável *a*

$a = b = c = d = 0$
a,b=b,a
a,*b=lista
a+=1

Matemática

```
a+b
a-b
a*b
a/b
a//b
a**b
a%b
abs(a)
sin(a)
sqrt(a)
log(a)
ceil(a)
floor(a)
```

```
relacionais: <, >, <=, <=, ==, !=
             in,

lógicos: and, or e not
```

Entrada e Saída

- Entrada

```
a = int(input("Forneça o valor "))
```

  - Neste comando, a função pára
  - Mostra a mensagem ("Forneça o valor ")
  - Aguarda o operador digital algo
  - o que for digitado é aceito como STRING
  - neste exemplo é transformado em inteiro
  - jogado na variável *a*

- Saída

```
print(expressao, variável, exp, var, ...)
    exemplos:
    print (a) # o valor de a é impresso
    print ("o valor de a é = ",a)
    print (a,b,c,d,e)
    print ("a=",a," b=",b," c=",c," d=",d)
```

Importação de módulos

- 170.000 módulos
- alguns muito grandes
- só se carrega o que se vai usar

```
import aaa
import aaa as b
from aaa import *
```

Programação

- Pode ser usado como script ou como programa
- execução via F5 do editor
- ou chamar python função
- Interpretado, logo não muito eficiente

```
defun a(parâmetros):
    ...
    ...
    return ...
acabou a função a
```

Exemplo

```
graos=0
for i in range(1,65):
  graos=graos+2**i
print('graos',graos)
kilos=graos/170000
```

ou

```
def dist(x,y):
  su=(x[1]-y[1])**2
  su=su+(x[2]-y[2])**2
  su=su+(x[3]-y[3])**2
  su=su+(x[4]-y[4])**2
  su=su+(x[5]-y[5])**2
  su=su+(x[6]-y[6])**2
  su=su**0.5
  return su
```

Repetições

```
while <condição>
    ...bloco

for <interador>
    ...bloco

    break

    continue
```

Interadores

Em python o interador está para o indexador em C++ Mas, é muito mais do que isso: elementos em lista, chaves em dicionário, linhas em relatório objetos em classe, registros em um arquivo,...

```
a={1:300,5:400,7:500}
for i in a:         # impressos:
  print(i)          # 1, 5 e 7
a=(1,20,300)
for i in a:         # impressos:
  print(i)          # 1, 20 e 300
a={1,6,8}
for i in a:         # impressos:
  print(i)          # 1, 6 e 8
a=[1, 6, 8, 9, 11, 20]
print([x for x in a if x%2==0])
    # impressos [6,8,20]
[x**2 for x in range(5)]
```

Operações

tamanho	`len(c)`	len([1,2,3])→ 3
mínimo	`min(c)`	min([3,2,7])→ 2
máximo	`max(c)`	max([3,2,7])→ 7
soma	`sum(c)`	sum([1,2,3])→ 6
sort	`sort(c)`	sort([3,2,7])→[2,3,7]
pertença	`x in c`	3 in [3,2,7]→ True
todos	`all(c)`	verdadeiro se todos os *c* são verdadeiros
algum	`any(c)`	verdadeiro se algum *c* é verdadeiro
reverso	`reversed(c)`	*c* em ordem reversa
posição	`c.index(v)`	índice de *v* em *c*
contador	`c.count(v)`	quantas ocorrências de *v* em *c*

Ops em listas

inserção	`L.append(v)`	insere v ao final de L
inserçao múltipla	`L.extend(p)`	insere a lista p no final de L
inserção	`L.insert(i,v)`	insere v após o índice i em L
remoção	`L.remove(v)`	remove a 1^o ocorrência de v em L
pop	`L.pop([i])`$\rightarrow$ v	remove e retorna $L[i]$
ordena	`L.sort()`	ordena *in place*
reverte	`L.reverse()`	reverte *in place*

ops em dicionários

chave	`d[chave]=v`	atualiza a chave de *d* com *v*
chave	`d[chave]` $\rightarrow$v	consulta a chave de *d*
limpa	`D.clear()`	limpa o dicionário *D*
exclui	`del D[chave]`	elimina a entrada *chave*

```
def fibo(n):
  if n<3:
    return 1
  else:
    return fibo(n-1)
   +fibo(n-2)
xx = int(input
('informe n '))
print(fibo(xx))
```

```
d = {1:1, 2:1}
def fibm(n):
  aa=d.get(n)
  if aa is None:
     d[n]=fibm(n-1)+fibm(n-2)
 return d[n]
   else:
      return aa
xx = int(input('Informe n '))
print(fibm(xx))
```

Ops em conjuntos

união	|	a=$\{1,2,3\}$,b=$\{4,6,9\}a\|b \rightarrow\{1,2,3,4,6,9\}$
intersecção	&	a=$\{1,4,9\}$, b=$\{4,6,9\}a\|b \rightarrow\{4,9\}$
diferença	-	a=$\{1,4,5,6,8,9\}$, b=$\{2,3,4,5\}$,$a - b \rightarrow \{8,1,6,9\}$
diferença simétrica	^	a^b$\rightarrow \{1,2,3,6,8,9\}$
adição	`S.add(v)`	Adiciona elemento v no dicionário S
exclusão	`S.remove(v)`	remove v no dicionário S
limpeza	`S.clear()`	zera o dicionário S

Ops em arquivos

abertura	`f=open(a,t)`	abre o arquivo *a* e cria a variável *f*. *t* pode ser *w*, *r* ou *a*
leitura	`a=f.read(n)`	Lê *n* caracteres de *f* e joga em *a*
escrita	`f.write(q)`	grava *q* no arquivo *f*
le linha	`f.readline()`	lê a próxima linha em *f*
le linhas	`f.readlines(n)`	lê *n* próximas linhas em *f*
fechamento	`f.close()`	Fecha o arquivo *f*
aplica	`f.flush()`	aplica mudanças em *f*

Ops em strings

S.split([e])	Cria lista, separando pelo *e* [espaços]
[j].join(L)	junta elementos de *L*, conectando-os com *j*
S.strip([c])	Remove *c* ou brancos no início e fim de *S*
S.count('c')	Conta quantos *c* há em *S*
S.partition(e)	Separa em 3 pedaços: antes, *e* e depois
S.find(u,i,f)	Retorna o primeiro *u* começando em *i*, terminando em *f* ou -1
S.is...()	... pode ser alpha... Retorna True se sim
S.upper()	Retorna maiúsculas
S.lower()	Retorna minúsculas
S.title()	Só as iniciais maiúsculas
S.swapcase()	troca a caixa
S.center(t[,f])	centraliza em *t* preenchendo com *f*

Classes

```
class Carro:
  def __init__(self, marca, ano):
      #construtor
    self.marca=marca
    self.ano=ano
  def metodo(self):
    print("a marca e ",self.marca)
class Colecao(Carro):
     # colecao herda Carro
  def __init__(self,m,a,c):
    Carro.__init__(self,m,a)
    self.cor=c
c1=Carro("fusca",1970)
c1.metodo()
c2=Colecao("chevete",1973,"branco")
```