

Python - Minecraft

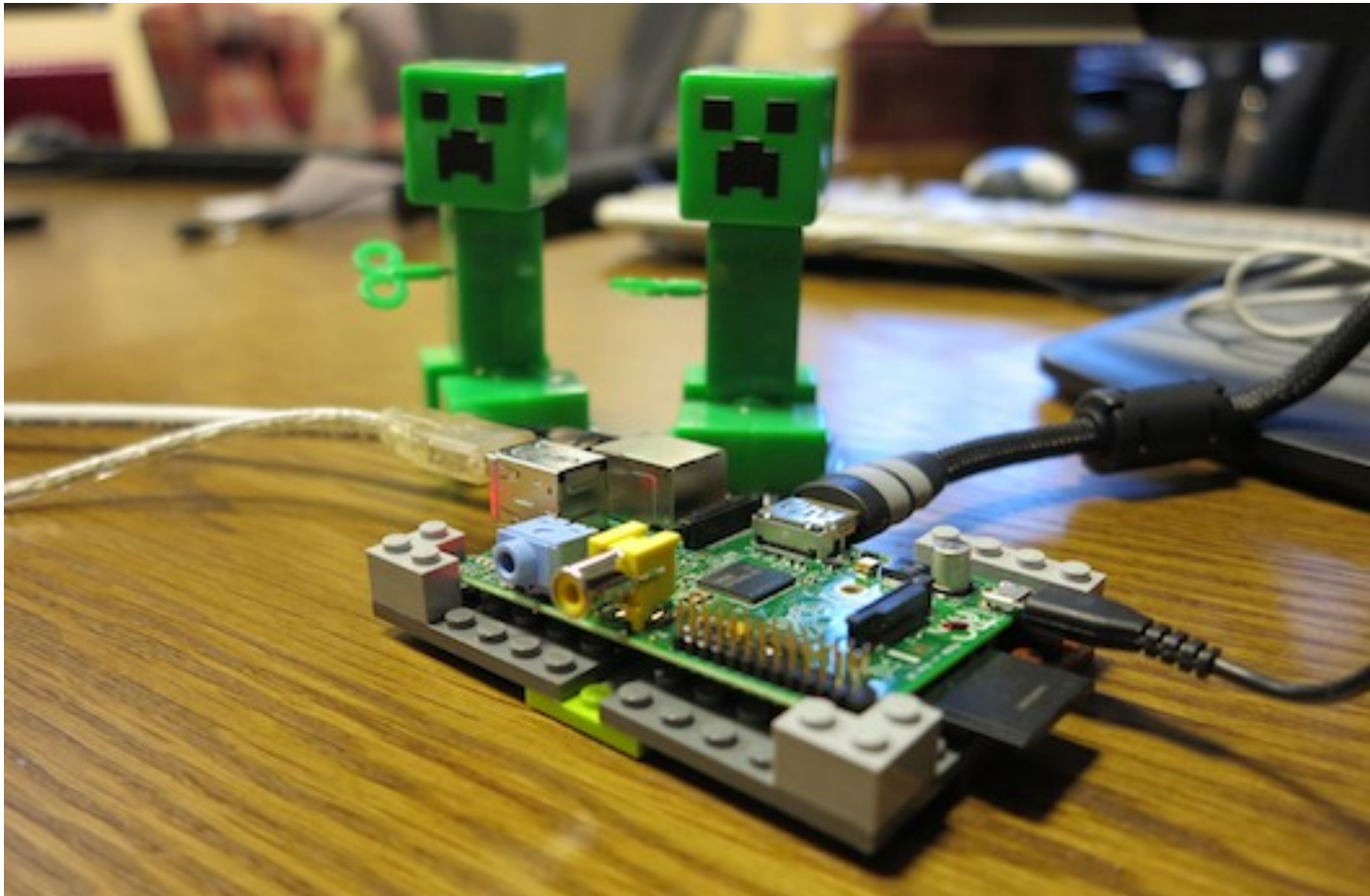


Origens

Minecraft



Raspberry PI



Minecraft PI Edition

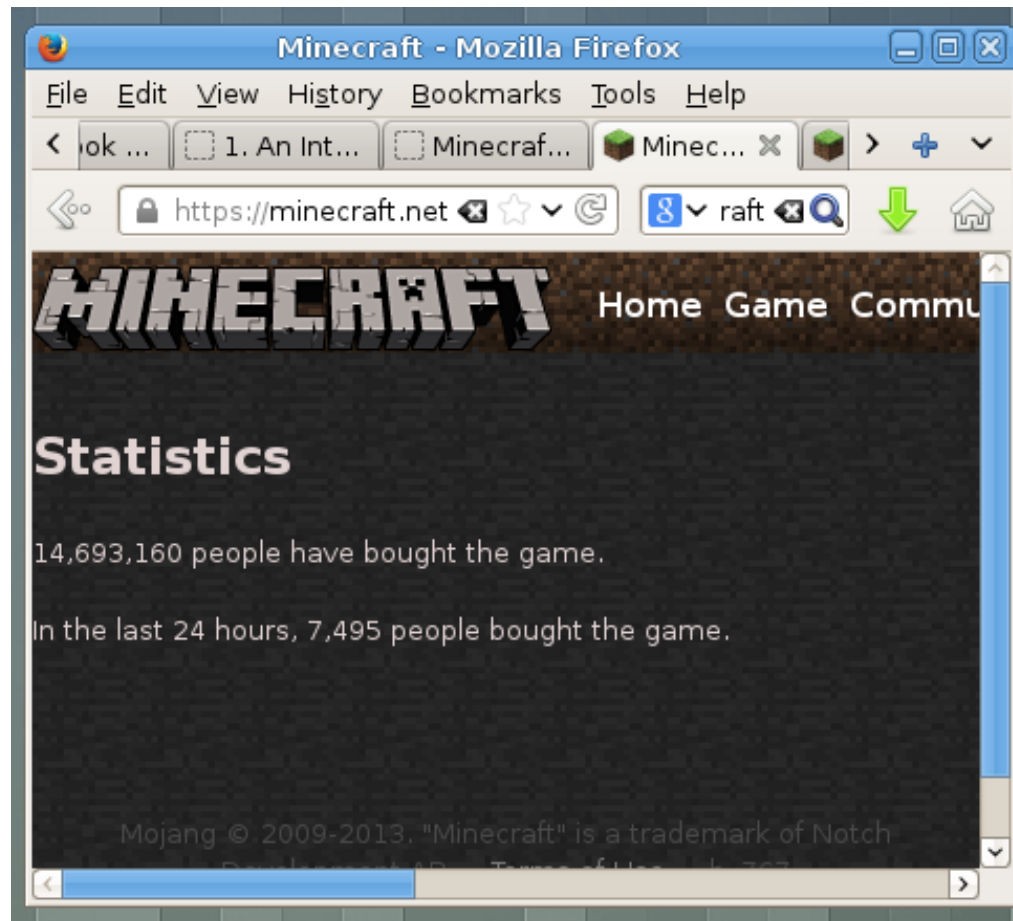


Python

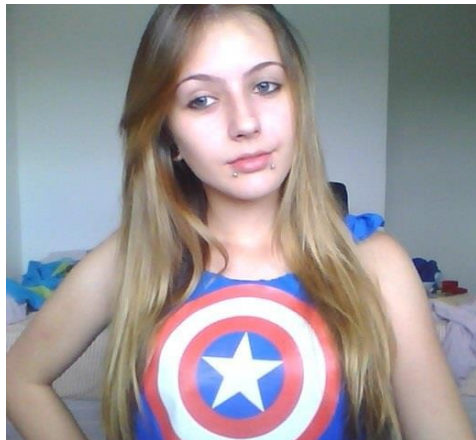


Motivações

Popularidade



Público Alvo



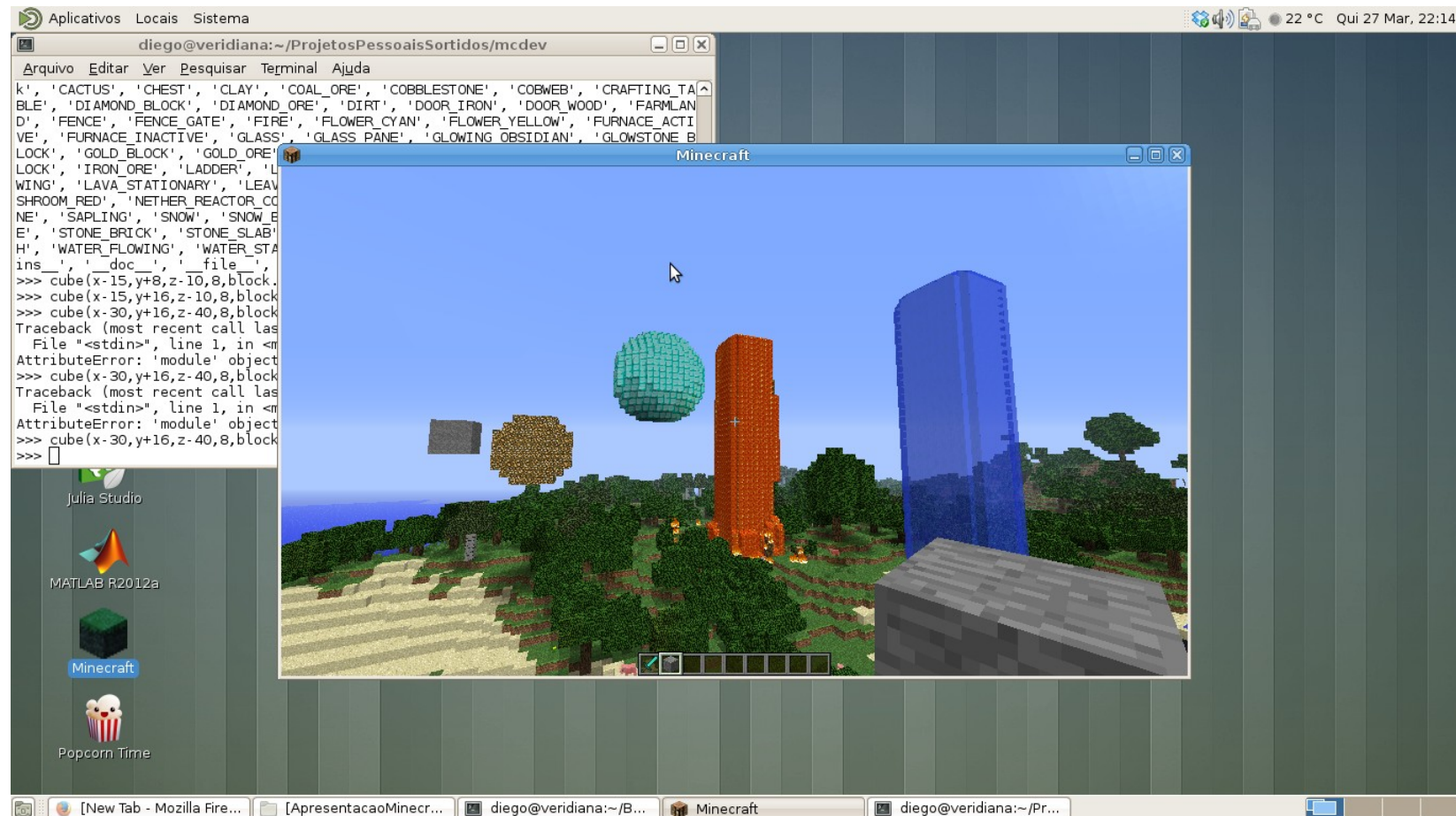
Diversão



Diversão e Aplicação



Concretização de Abstrações



Ferramentas

Minecraft 1.6.4



Bukkit Server 1.6.4



MCPI

```
pi@raspberrypi ~ $ wget https://s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz
--2013-10-01 09:13:57-- https://s3.amazonaws.com/assets.minecraft.net/pi/minecraft-pi-0.1.1.tar.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 205.251.243.82
Connecting to s3.amazonaws.com (s3.amazonaws.com)|205.251.243.82|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1459472 (1.4M) [application/x-gzip]
Saving to: `minecraft-pi-0.1.1.tar.gz'

100%[=====>] 1,459,472    538K/s   in 2.6s

2013-10-01 09:14:07 (538 KB/s) - `minecraft-pi-0.1.1.tar.gz' saved [1459472/1459472]

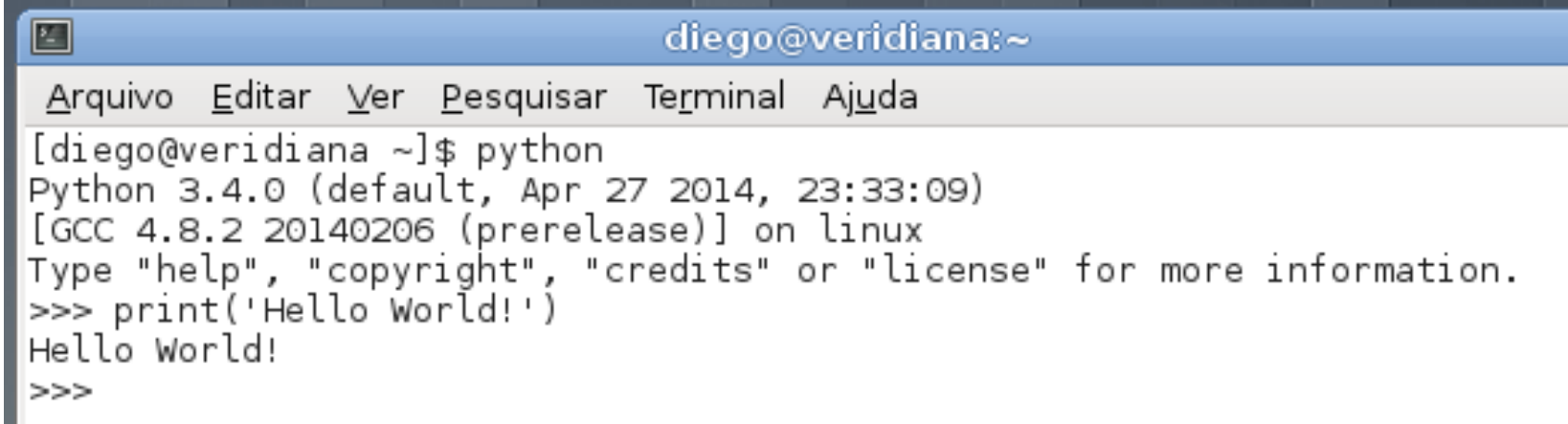
pi@raspberrypi ~ $ tar -zxvf minecraft-pi-0.1.1.tar.gz
mcpi/
mcpi/api/
mcpi/CONTROLS.txt
mcpi/data/
mcpi/HOW_TO_RUN.txt
mcpi/LICENSE.txt
mcpi/minecraft-pi
mcpi/VERSION.txt
mcpi/data/images/
```

Raspberry Juice 1.3



Conceitos - Python

Hello World

A screenshot of a terminal window titled 'diego@veridiana:~'. The window has a menu bar with options: 'Arquivo', 'Editar', 'Ver', 'Pesquisar', 'Terminal', and 'Ajuda'. The terminal content shows the execution of the Python command 'python'. It displays the Python version '3.4.0 (default, Apr 27 2014, 23:33:09)', the GCC version '4.8.2 20140206 (prerelease)', and the operating system 'linux'. It then prompts the user to type 'help', 'copyright', 'credits', or 'license' for more information. The user enters '>>> print('Hello World!')', and the terminal outputs 'Hello World!'. The prompt '>>>' is shown again at the bottom.

```
diego@veridiana:~  
Arquivo  Editar  Ver  Pesquisar  Terminal  Ajuda  
[diego@veridiana ~]$ python  
Python 3.4.0 (default, Apr 27 2014, 23:33:09)  
[GCC 4.8.2 20140206 (prerelease)] on linux  
Type "help", "copyright", "credits" or "license" for more information.  
>>> print('Hello World!')  
Hello World!  
>>>
```

Declaração e acesso a Variáveis

```
>>> #String:
...
>>> a = 'string'
>>> a
'string'
>>> #Inteiro:
... a = 1
>>> a
1
>>> #Flutuante:
... a = 0.1
>>> a
0.1
>>> #Vetores:
... a = [0, 1, 2]
>>> a[::-1]
[2, 1, 0]
>>> a
[0, 1, 2]
>>>
```


Declaração e Acesso a Variáveis

```
# Booleano:
... a = True
>>> a
True
>>> a = False
>>> a
False
>>> # Recebendo valores da entrada padrão (teclado)
... a = input('Entre 1 número')
Entre 1 número 4
>>> a
'4'
>>> a = input('Os valores da entrada padrão são string por default?: ')
Os valores da entrada padrão são string por default?: 5
>>> a
'5'
>>> isinstance(a, str) #str é o tipo padrão de string em python
True
>>> # Python é fortemente tipada, apesar de ser dinamica
... a + 4
Traceback (most recent call last):
  File "<stdin>", line 2, in <module>
TypeError: Can't convert 'int' object to str implicitly
>>> # precisa de converter a em inteiro para operar a soma
... a = int(a)
>>> a + 4
9
>>>
```

Operações:

```
>>> #Soma:
... 1 + 1
2
>>> #Subtração:
... 1 - 1
0
>>> #Multiplicação:
... 2 * 2
4
>>> #Divisão:
... 2 / 2
1.0
>>> #Potência:
... 2 ** 2
4
>>> 4 ** (1/2)
2.0
```

Operações:

```
#Não existe ++
... a = 1
>>> a ++
    File "<stdin>", line 1
      a ++
        ^
SyntaxError: invalid syntax
>>> #Python tudo deve ser explíci
... a += 1
>>> a
2
>>> # O mesmo vale ao menos
... a --
    File "<stdin>", line 2
      a --
        ^
SyntaxError: invalid syntax
>>> a -= 1 #Explicitando
>>> a
1
—
```

Avaliação de Condições

```
>>> # Igualdade
... 1 == 1
True
>>> # Não Igualdade
... 1 != 2
True
>>> # Desigualdade menor (menor igual)
... 1 <= 2
True
>>> # Desigualdade maior (maior igual)
... 2 >= 3
False
>>> # Negação (Inversão)
... a = True
>>> not a
False
>>> # Ou
... a or 1 != 2
True
>>> # E
... a and 1 != 2
True
>>> a and 1 == 2
False
```

PEP 8



PEP 8

- Guia de boas práticas de Python
- O código em Python deve ser lindo, limpo e explícito, facilitando o entendimento
- Um bloco de execução tem seu escopo delimitado pela indentação (Não existem chaves)
- Zen of Python

PEP 8

```
# Errado:
... if 1 == 1:
...     print('1 é igual a 1')
      File "<stdin>", line 3
        print('1 é igual a 1')
        ^
IndentationError: expected an indented block

>>> # Certo:
... if 1 == 1:
...     print('1 é igual a 1')
...
1 é igual a 1
>>> █
```

- O padrão no PEP8 é que a indentação tenha 4 espaços em branco

Estruturas de Controle

```
a = input('Entre um numero entre 1 e 4: ')
Entre um numero entre 1 e 4: 5
>>> if int(a) < 1:
...     print('Erro, seu número está abaixo do limite inferior do intervalo')
... elif int(a) <= 4:
...     print(a)
... else:
...     print('Erro, seu número está acima do limite superior do intervalo')
...
Erro, seu número está acima do limite superior do intervalo
>>>
```

Estruturas de Repetição - For

```
# Estrutura de Repetição - For
... # Acessando elementos de um vetor:
... vec = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
>>> for e in vec:
...     print(e)
...
0
1
2
3
4
5
6
7
8
9
>>> # Em python existe somente esse tipo de for (for each)
... # Para se utilizar o for indo em intervalos, usa-se a função range
... for i in range(0,10):
...     print(i)
...
0
1
2
3
4
5
6
7
8
9
>>>
```

Estruturas de Repetição - While

```
# Como Python possui somente o for each, para se repetir a execução
... # de um bloco de código até determinada condição ser atingida,
... # é recomendável utilizar-se do while
... # WHILE:
... i = 0
>>> while i < 10:
...     print(i)
...     i += 1
...
0
1
2
3
4
5
6
7
8
9
>>> while True:
...     pass
...

^CTraceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyboardInterrupt
>>> while False:
...     print('Não vai printar')
...
>>>
```

Funções e Procedimentos

```
>>> # Funções e Procedimentos
... # para se criar funções e procedimentos em python, utiliza-se da palavra
... # reservada def, em seguida o nome da funcao / procedimento e seus para-
... # metros.
... # Procedimento (sem retorno):
... def soma_dois_numeros():
...     numero_1 = int(input('Entre o primeiro numero: '))
...     numero_2 = int(input('Entre o segundo numero: '))
...     print('Soma dos dois numeros: %d' %(numero_1+numero_2))
...
>>> soma_dois_numeros()
Entre o primeiro numero: 4
Entre o segundo numero: 5
Soma dos dois numeros: 9
>>> # Procedimento com parametros:
... def soma_dois_numeros(a, b):
...     print('Soma dos dois numeros: %d' %(a+b))
...
>>> soma_dois_numeros(3,4)
Soma dos dois numeros: 7
>>> # Procedimentos com parametros - usando valores default nos parametros
>>> def soma_dois_numeros(a=0, b=0):
...     print('Soma dos dois numeros: %d' %(a+b))
...
>>> soma_dois_numeros()
Soma dos dois numeros: 0
>>> soma_dois_numeros(1)
Soma dos dois numeros: 1
>>> soma_dois_numeros(4)
Soma dos dois numeros: 4
>>> soma_dois_numeros(4, 1)
Soma dos dois numeros: 5
```

Funções e Procedimentos

```
# Funções - procedimentos com retorno
... # Função de parâmetros com valor default
... def soma_dois_numeros(a=0, b=0):
...     return a + b
...
>>> soma_dois_numeros()
0
>>> soma_dois_numeros(1)
1
>>> soma_dois_numeros(3)
3
>>> soma_dois_numeros(3, 1)
4
>>> soma_dois_numeros(4, 1)
5
>>> # Função de somente o primeiro parametro com valor default
... def f (a=1, b):
...     return a * b
...
File "<stdin>", line 2
SyntaxError: non-default argument follows default argument
>>> # Dá bizil, como podem ver, parametros com valor default
... # não podem preceder parâmetros sem valor default.
... # O jeito certo:
... def f(a, b=1):
...     return a*b
...
>>> f(3)
3
>>> f(3,3)
9
... f(3, 3)
```

Sobrescrevendo Operadores

```
# Sobrescrevendo Operadores
... class int42(int): # Aqui criamos uma subclasse da classe int
...     def __add__(self, other): # self objeto corrente, other - a ser somado
...         return 42 # qualquer soma entre objetos da classe int42 retorna 42
...
>>> b = int42(4)
>>> b
4
>>> c = int42(5)
>>> c
5
>>> b+c
42
>>> b-c
-1
>>> b+c
42
>>> # Duck typing em python, posso somar b a um int normal sem viadagem?
... b + 1
42
>>> # Posso, Python FTW
...
>>> # JAVA SUCKS
...
>>>
```

Exemplos - Minecraft

Hello World



Hello World

```
import mcpi.minecraft as minecraft  
mc = minecraft.Minecraft.create()  
mc.postToChat('Hello Minecraft World!')
```

Bloco



Bloco

```
import mcpi.minecraft as minecraft
import mcpi.block as block
mc = minecraft.Minecraft.create()

# coloca um bloco de pedra 3 blocos acima da cabeça do jogador
[x,y,z] = mc.player.getPos()
mc.setBlock(x,y+3,z,block.STONE)
```

Linha e Parede



```
import mcpi.minecraft as minecraft
import mcpi.block as block
mc = minecraft.Minecraft.create()

# desenha uma linha de 5 blocos de pedra à frente do jogador
[x,y,z] = mc.player.getPos()
i = 0
while i < 5:
    mc.setBlock(x+i,y,z+3,block.STONE)
    i += 1
```



```
# desenha uma parede de 5 x 8 à frente do jogador
[x,y,z] = mc.player.getPos()
i = 0
while i < 5:
    j = 0
    while j < 8:
        mc.setBlock(x+i,y+j,z+3,block.STONE)
        j += 1
    i += 1
```

Sólidos - Cubo



Cubo

```
def cube(x, y, z, side, material):  
    for i in range (0,side):  
        for j in range (0,side):  
            for k in range (0,side):  
                mc.setBlock(x+i,y+j,z+k,material)
```

Sólidos - Esfera



Alguns Desafios

- Buildings em geral
- Redstone Circuitry
- Interação Player – Server – Chat
- Integração Minecraft / Mundo Real

Complementações

- Professor Fernando Masanori - <https://twitter.com/fmasanori>
- Python pra zumbis - <http://pycursos.com/python-para-zumbis/>
- mcpi.py - <http://mcpi.py.wordpress.com/>

The End



Referências

- Minecraft 1.6.4 (Disponível no Launcher)
- Minecraft Pi Edition:
<http://pi.minecraft.net/>
- Pi Programming – Minecraft: <http://tinyurl.com/introducao-minecraft-pi>
- Raspberry Juice + MCPI:
<http://dev.bukkit.org/bukkit-plugins/raspberryjuice/files/4-raspberry-juice-v1-3/>
- Bukkit Server 1.6.4:
https://dl.bukkit.org/downloads/bukkit/get/02388_1.6.4-R2.0/bukkit.jar
- Apresentação e Códigos dos Exemplos:
<https://github.com/DiegoAscanio/xisto>