Linguagem Python

Prof. Rodrigo Macedo

Escopo do Curso

- Conceitos Iniciais
- Instalação e Configuração
- Tipos de Dados
- Estrutura de Dados (Listas e Dicionários).
- Strings.
- Condicionais.
- Laço de Repetição.
- Funções.
- Questão de Concursos

Conceitos Iniciais

- Python é uma linguagem de programação de alto nível, interpretada, de script, imperativa, orientada a objetos, funcional, de tipagem dinâmica e forte. Foi lançada por Guido van Rossum em 1991.
- A linguagem foi projetada com a filosofia de enfatizar a importância do esforço do programador sobre o esforço computacional. Prioriza a legibilidade do código sobre a velocidade ou expressividade. Combina uma sintaxe concisa e clara com os recursos poderosos de sua biblioteca padrão e por módulos e frameworks desenvolvidos por terceiros.

Conceitos Iniciais

- Python é uma linguagem de propósito geral de alto nível, multiparadigma, suporta o paradigma orientado a objetos, imperativo, funcional e procedural. Possui tipagem dinâmica e uma de suas principais características é permitir a fácil leitura do código e exigir poucas linhas de código se comparado ao mesmo programa em outras linguagens.
- O Python é uma linguagem de programação de uso geral. Atualmente, é uma das linguagens mais populares no mundo, sendo usada nos mais diversos campos, tais como machine learning, inteligência artificial, desenvolvimento de jogos, pesquisas acadêmicas e ensino de programação

Conceitos Iniciais

- Atualmente a linguagem está na versão 3.9.
- É multiplataforma.
- Blocos de códigos são delimitados por endentação.
- Palavras reservadas da linguagem.
- Interpretador interativo.

```
>>> 1+1
2
>>>
>>> a = 1+1
>>> print a
2
>>> print(a)
2
>>>
```

```
False      None     True     and      as
assert     break     class     continue
def    del    elif     else     except
finally     for     from     global     if
import     in     is     lambda     not
nonlocal     or     pass     raise     try
return     while     with     yield
```

Endentação


Programa
Instruções
Enquanto condição:
Instruções
Se condição:
Instruções
Senão:
Instruções
Instruções
Inicio do programa
Dois pontos obrigatórios
Inicio de um bloco
Outro bloco
Fim dos dois blocos
Fim do programa

Exemplo – Shell Interativo

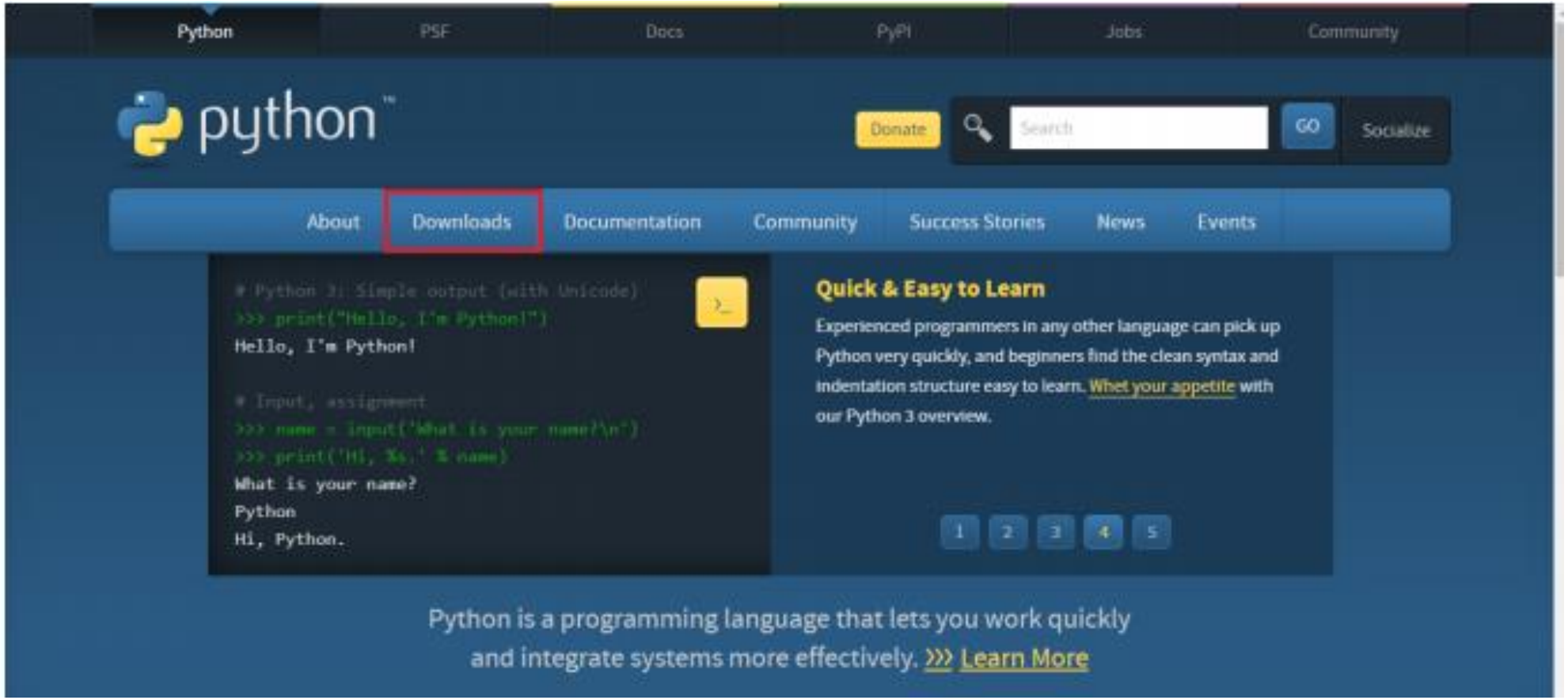
```
C:\WINDOWS\system32>python
Python 3.6.5 |Anaconda, Inc.| (default, Mar 29 2018, 13:32:41) [MSC v.1900 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello WOrld")
Hello WOrld
>>> _
```

Instalação Python

- Há duas formas de instalar o Python: você pode baixar o interpretador independente, ou você pode baixar o Anaconda, que é uma stack de desenvolvimento em Python, que além de vir com o interpretador em Python, vem com muitos outras bibliotecas instaladas.
- Nos dois casos, terá o gerenciador de bibliotecas python (Pip), que permite baixar bibliotecas via comando.
- Para instalar uma biblioteca, usa-se o comando pip install name.
- Todas as informações das bibliotecas do pip se encontram no site: https://pypi.org/

Instalação Python - Standalone

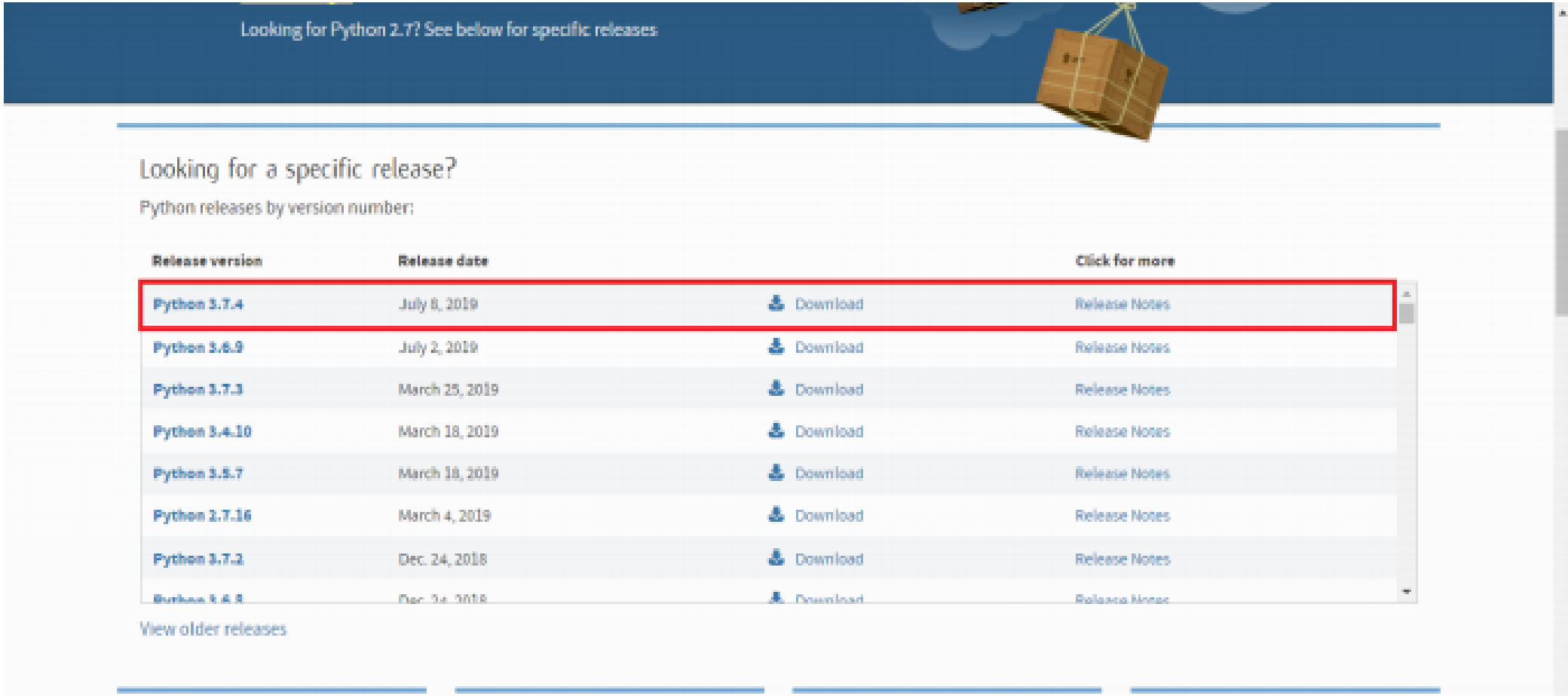
- Deve-se abrir o site oficial (https://www.python.org) e ir na seção Downloads.



Instalação Python - Standalone

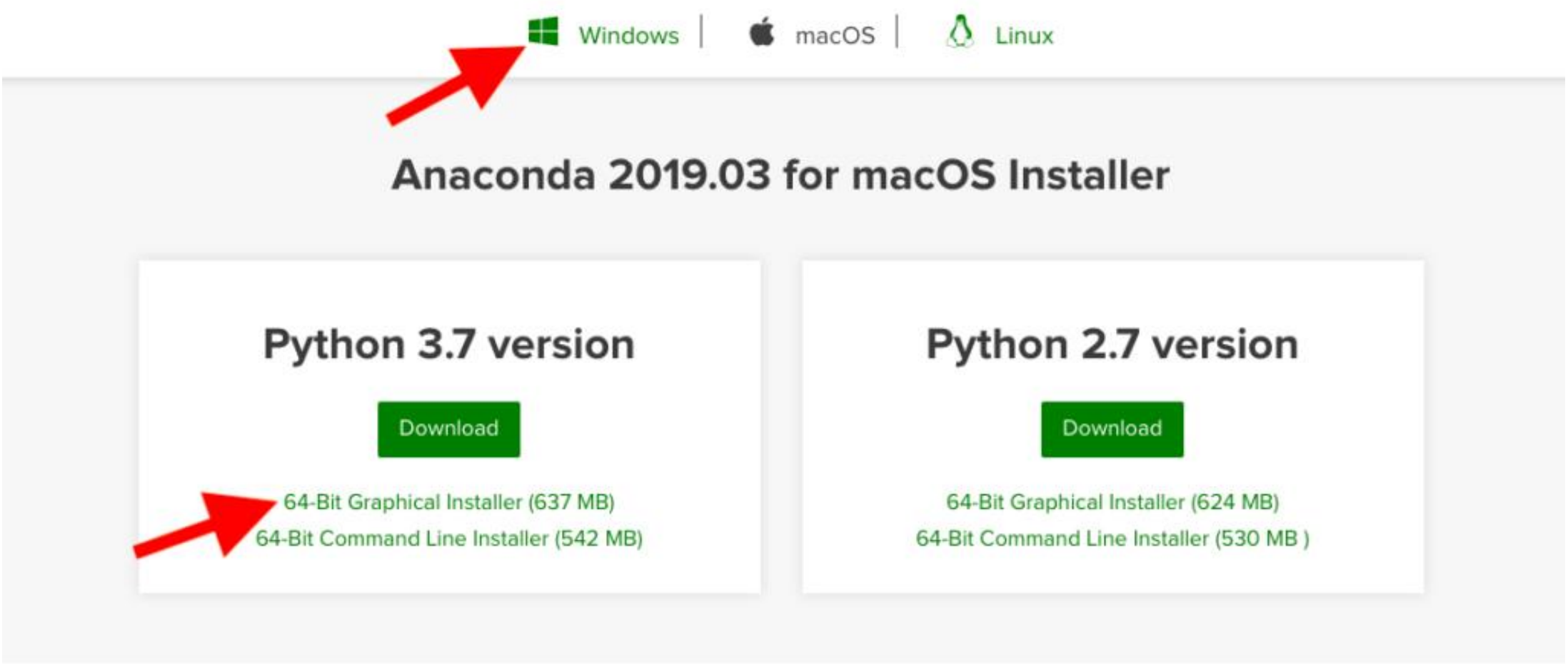
- Selecionar a versão e instalador adequado para o sistema.

Por fim, basta abrir o instalador e clicar em Install Now. É recomendado marcar as duas caixas de seleção para que também seja possível acessar o Python pelos terminais do Windows.



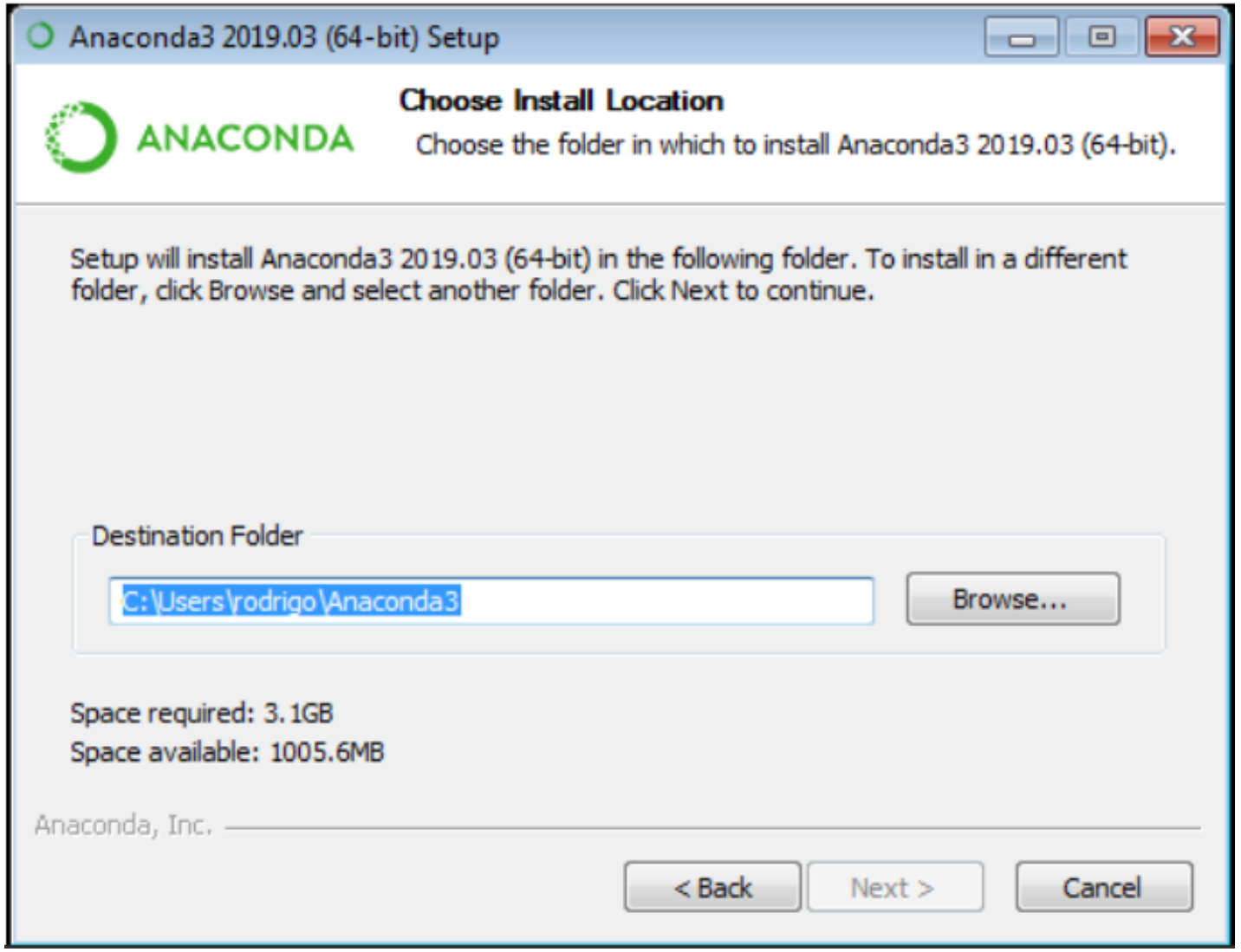
Instalação Python - Anaconda

- Basta acessar o site (https://www.anaconda.com/) e procurar a seção Downloads.



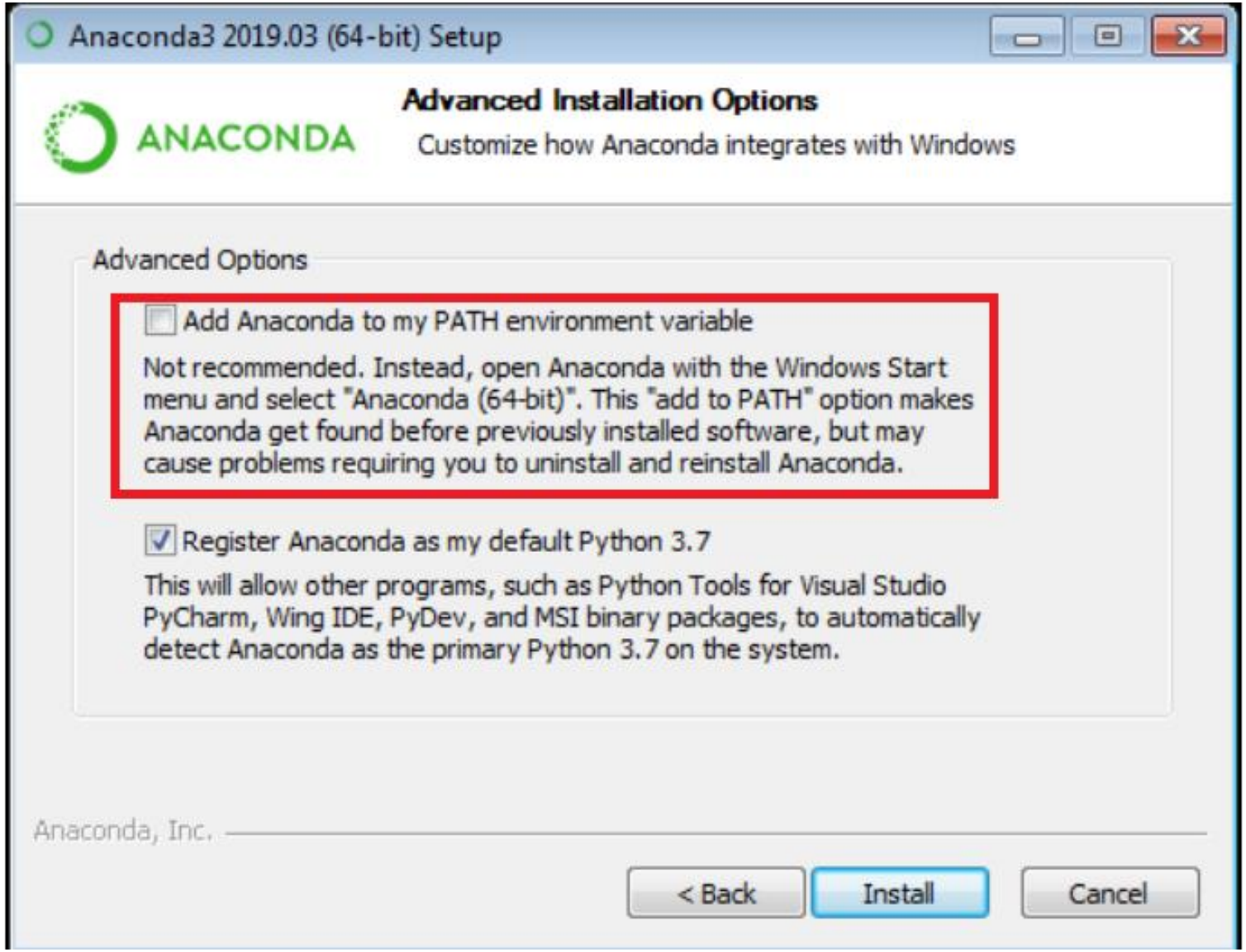
Instalação Python - Anaconda

- Por padrão o diretório de instalação será no diretório atual do usuário.

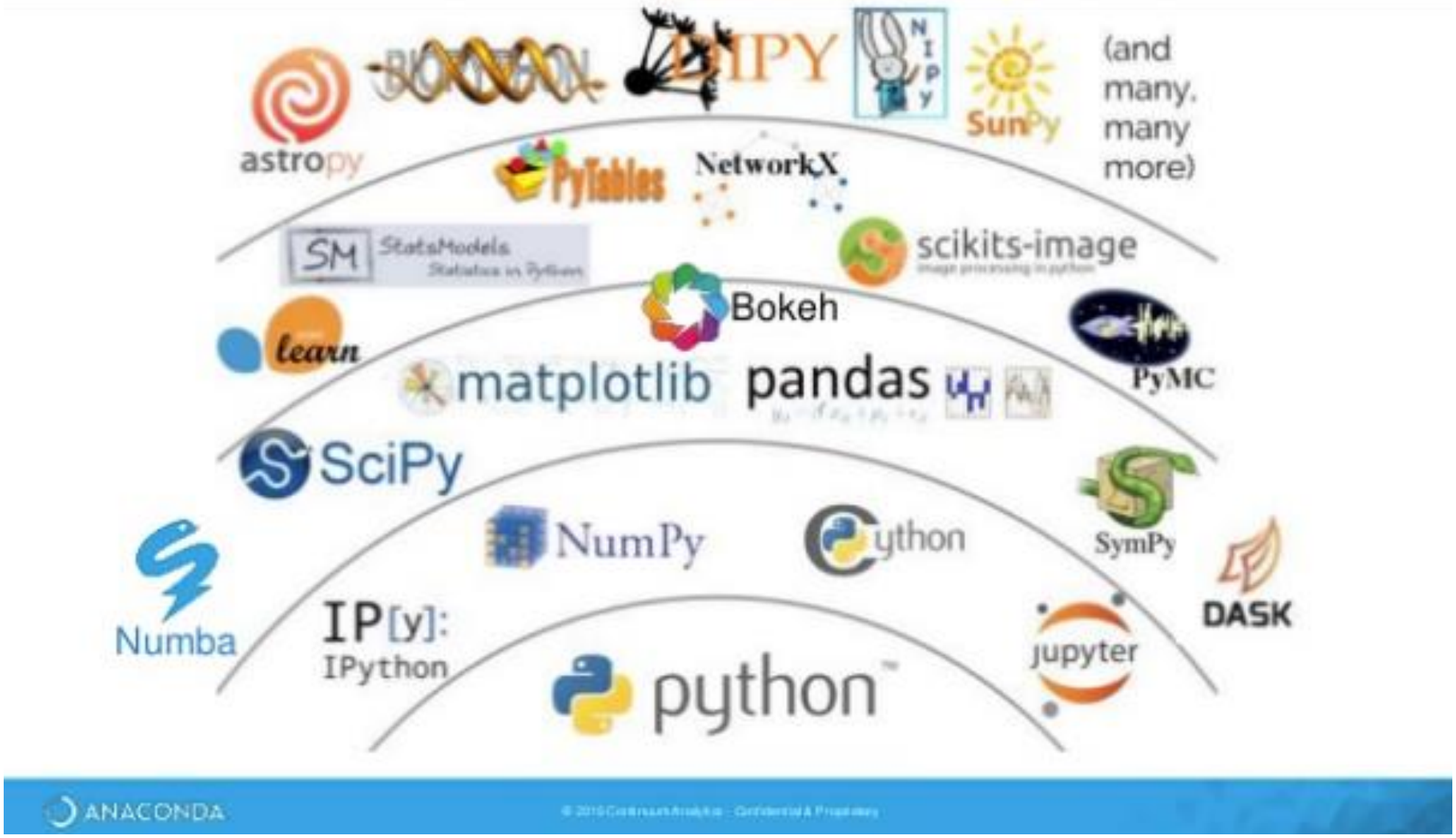


Instalação Python - Anaconda

- Interessante marcar a opção em vermelho para interagir com o python via CMD.



Instalação Python - Anaconda



Tipos de Dados: Numéricos

- Integer - valores inteiros.
- Float - valores decimais.

```
In [1]: type(5)
Out[1]: int

In [2]: type(5.0)
Out[2]: float
```

```
In [3]: # Soma
        4 + 4
Out[3]: 8

In [4]: # Subtração
        4 - 3
Out[4]: 1

In [5]: # Multiplicação
        3 * 3
Out[5]: 9
```

```
In [6]: # Divisão
        3 / 2
Out[6]: 1.5

In [7]: # Potência
        4 ** 2
Out[7]: 16

In [8]: # Módulo
        10 % 3
Out[8]: 1
```

```
In [10]: # Resultado é um número float
         4 / 2
Out[10]: 2.0

In [11]: # Resultado é um número inteiro
         4 // 2
Out[11]: 2

In [12]: 4 / 3.0
Out[12]: 1.3333333333333333

In [13]: 4 // 3.0
Out[13]: 1.0
```

```
In [14]: # Retorna o valor absoluto
         abs(-8)
Out[14]: 8

In [15]: # Retorna o valor absoluto
         abs(8)
Out[15]: 8

In [16]: # Retorna o valor com arredondamento
         round(3.14151922,2)
Out[16]: 3.14
```

```
In [17]: # Potência
         pow(4,2)
Out[17]: 16

In [18]: # Potência
         pow(5,3)
Out[18]: 125
```

```
In [1]: int(5.9)
Out[1]: 5

In [2]: int(4.2)
Out[2]: 4
```

Tipos de Dados: Lógico

```
In [2]: b , a = 20, 22

In [3]: b
Out[3]: 20

In [4]: a
Out[4]: 22
```

```
In [5]: b >= a
Out[5]: False
```

```
In [6]: a = True

In [7]: b = 20 ==10

In [8]: b
Out[8]: False

In [9]: a | b
Out[9]: True
```

Operador OU lógico

```
In [11]: a & b
Out[11]: False
```

Operador E lógico

Operadores Matemáticos

Operador	Descrição	Exemplo
+	Adição	2 + 3
-	Subtração	2 - 2
*	Multiplicação	2 * 3
/	Divisão	5/4
//	Divisão Inteira	4/3
**	Exponenciação	2**100
%	Resto de Divisão(mod)	10 % 2

Operadores Lógicos

Operação	Símbolo	Exemplo	Resultado
Igual	**=**	7 = 7	VERDADEIRO
Maior que	**>**	10 > 20	FALSO
Menor que	**<**	100 < 1000	VERDADEIRO
Menos ou igual a	**<=**	1.25 <= 2.50	VERDADEIRO
Maior ou igual a	**>=**	1234 >= 1234	VERDADEIRO
Diferente de	**<>**	10 <> 10	FALSO

Strings

```
In [1]: # Pode ser usado aspa simples
        'Criando uma string em Python'

Out[1]: 'Criando uma string em Python'
```

```
In [2]: # Podemos usar aspas duplas
        "Podemos usar aspas duplas ou simples para strings em Python"

Out[2]: 'Podemos usar aspas duplas ou simples para strings em Python'
```

```
In [3]: print ('Testando \nStrings \nem \nPython')

        Testando
        Strings
        em
        Python
```

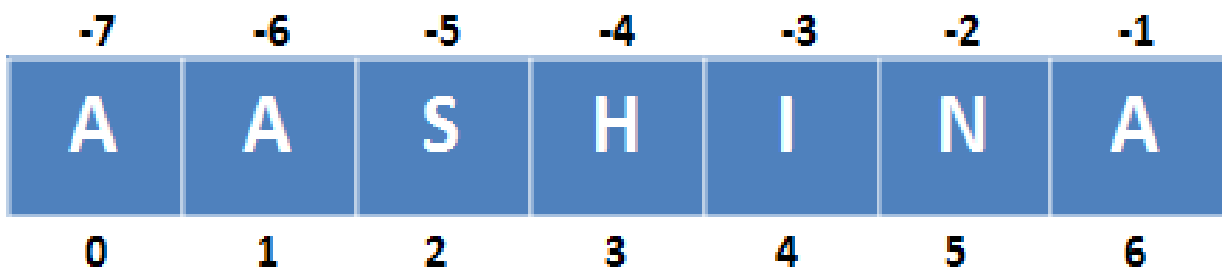
No Python, é possível usar Strings com aspas simples e / ou, com aspas duplas.

Strings - Indexação

```
In [8]: # Atribuindo uma string
        s = 'Provas de TI'

In [9]: print(s)
        Provas de TI
```

```
In [10]: # Primeiro elemento da string.
         s[0]
Out[10]: 'P'

In [11]: s[1]
Out[11]: 'r'

In [12]: s[2]
Out[12]: 'o'
```

STRING = "AASHINA"



STRING[0] = 'A'
STRING[1] = 'A'
STRING[2] = 'S'
STRING[3] = 'H'
STRING[4] = 'I'
STRING[5] = 'N'
STRING[6] = 'A'

STRING[-7] = 'A'
STRING[-6] = 'A'
STRING[-5] = 'S'
STRING[-4] = 'H'
STRING[-3] = 'I'
STRING[-2] = 'N'
STRING[-1] = 'A

Strings - Indexação e fatiamento

```
In [14]: s[0:]
Out[14]: 'Provas de TI'

In [15]: # Retorna tudo até a posição 3
         s[:3]
Out[15]: 'Pro'

In [16]: s[:]
Out[16]: 'Provas de TI'
```

```
In [23]: # Retorna tudo, exceto a última letra
         s[:-1]
Out[23]: 'Provas de T'

In [18]: # Fatiamento
         s[::1]
Out[18]: 'Provas de TI'

In [20]: # Fatiamento 2 em 2
         s[::2]
Out[20]: 'Poa eT'

In [21]: s[::-1]
Out[21]: 'IT ed savorP'
```

Strings - Outras operações

```
In [25]: # Concatenando strings
         s + ' é a melhor plataforma de preparação de concursos em TI'

Out[25]: 'Provas de TI é a melhor plataforma de preparação de concursos em TI'

In [26]: print(s)

         Provas de TI
```

Strings são imutáveis

```
In [36]: # Podemos usar o símbolo de multiplicação para criar repetição!
         palavra = 'concursos'

In [37]: palavra * 3

Out[37]: 'concursosconcursosconcursos'
```

Strings - Outras operações

```
In [38]: s
Out[38]: 'Provas de TI'

In [39]: # Upper Case
         s.upper()
Out[39]: 'PROVAS DE TI'

In [40]: # Lower case
         s.lower()
Out[40]: 'provas de ti'

In [41]: # Dividir uma string por espaços em branco (padrã
         s.split()
Out[41]: ['Provas', 'de', 'TI']
```

```
In [46]: s = 'seja bem vindo ao universo de python'

In [47]: s.capitalize()
Out[47]: 'Seja bem vindo ao universo de python'

In [48]: s.count('a')
Out[48]: 2

In [49]: s.find('p')
Out[49]: 30
```

```
In [50]: print("Python" == "R")
         False
```

```
In [53]: print("Python" == "Python")
         True
```

Strings - Placeholder x Format

```
In [59]: print("Meu nome é %s. Minha idade é %d" %("Fulano", 28))
```

```
Meu nome é Fulano. Minha idade é 28
```

```
In [65]: print("Meu nome é {fname}, Minha idade é {age}".
               format(fname = "Fulano", age = 28))
```

```
Meu nome é Fulano, Minha idade é 28
```

Strings - Outras operações

```
In [66]: print(" ".join(s))
s e j a   b e m   v i n d o   a o   u n i v e r s o   d e   p y t h o n
```

```
In [68]: print(" ".join(reversed(s)))
n o h t y p   e d   o s r e v i n u   o a   o d n i v   m e b   a j e s
```

```
In [72]: s.replace("python", "r")
Out[72]: 'seja bem vindo ao universo de r'
```

```
In [71]: print(", ".join(["Java", "Python", "R"]))
Java, Python, R
```

```
In [1]: txt = "Python é muito legal."

        x = txt.startswith("Python")

        x
Out[1]: True
```

Strings - Outras operações

```
In [73]: len(s)
```

```
Out[73]: 36
```

```
In [76]: busca = 'python'
         print(s.find(busca))
```

```
30
```

```
In [80]: print(s.title())
```

```
Seja Bem Vindo Ao Universo De Python
```

Listas

- Utiliza valores entre colchetes.

```
In [1]: # Criando uma lista
        listadomercado = ["ovos, farinha, leite, maças"]

In [2]: # Imprimindo a lista
        print(listadomercado)

['ovos, farinha, leite, maças']

In [3]: # Criando lista
        lista3 = [12, 100, "Universidade"]

In [4]: # Imprimindo
        print(lista3)

[12, 100, 'Universidade']
```

```
In [5]: # Atribuindo cada valor da lista a uma variável.
        item1 = lista3[0]
        item2 = lista3[1]
        item3 = lista3[2]

In [6]: # Imprimindo as variáveis
        print(item1, item2, item3)

12 100 Universidade
```

Listas - Indexação

```
In [8]: # Imprimindo um item da lista
        lista3[2]
Out[8]: 'Universidade'

In [9]: # Atualizando um item da lista
        lista3[2] = "chocolate"

In [10]: # Imprimindo lista alterada
         lista3
Out[10]: [12, 100, 'chocolate']
```

```
In [11]: # Deletando um item específico da lista
         del lista3[2]

In [13]: # Imprimindo o item com a lista alterada
         lista3
Out[13]: [12, 100]
```

Listas - Concatenação

```
In [14]: lista_s1 = [34, 32, 56]

In [15]: lista_s1
Out[15]: [34, 32, 56]

In [16]: lista_s2 = [21, 90, 51]

In [17]: lista_s2
Out[17]: [21, 90, 51]
```

```
In [18]: # Concatenando Listas
         lista_total = lista_s1 + lista_s2

In [19]: lista_total
Out[19]: [34, 32, 56, 21, 90, 51]
```

Listas - Outras operações

```
In [20]: # Criando uma lista
         lista_teste_op = [100, 2, -5, 3.4]

In [21]: # Verificando se o valor 10 pertence a lista
         print(10 in lista_teste_op)

         False

In [22]: # Verificando se o valor 100 pertence a lista
         print(100 in lista_teste_op)

         True
```

```
In [23]: len(lista_teste_op)
Out[23]: 4

In [24]: max(lista_teste_op)
Out[24]: 100

In [25]: min(lista_teste_op)
Out[25]: -5
```

Listas - Outras operações

```
In [26]: # Criando uma lista
         listadomercado2 = ["ovos", "farinha", "leite", "maças"]

In [27]: # Adicionando um item à lista
         listadomercado2.append("carne")

In [28]: listadomercado2

Out[28]: ['ovos', 'farinha', 'leite', 'maças', 'carne']

In [29]: listadomercado2.append("carne")

In [30]: listadomercado2

Out[30]: ['ovos', 'farinha', 'leite', 'maças', 'carne', 'carne']

In [31]: listadomercado2.count("carne")

Out[31]: 2
```

Listas - Outras operações

```
In [32]: x = [3, 4, 2, 1]

In [33]: x
Out[33]: [3, 4, 2, 1]

In [34]: # Ordena a lista
         x.sort()

In [35]: x
Out[35]: [1, 2, 3, 4]
```

```
In [37]: cidades
Out[37]: ['Recife', 'Manaus', 'Salvador', 'Fortaleza', 'Palmas']

In [38]: cidades.insert(2, 110)

In [39]: cidades
Out[39]: ['Recife', 'Manaus', 110, 'Salvador', 'Fortaleza', 'Palmas']
```

Dicionários

- Formado por pares de chave e valor e utiliza chaves.

```
In [3]: # Isso é um dicionário
        estudantes_dict = {"Mateus":24, "Fernanda":22, "Tamires":26, "Cristiano":25}

In [4]: estudantes_dict
Out[4]: {'Mateus': 24, 'Fernanda': 22, 'Tamires': 26, 'Cristiano': 25}

In [6]: estudantes_dict["Pedro"] = 23

In [7]: estudantes_dict["Pedro"]
Out[7]: 23

In [8]: estudantes_dict
Out[8]: {'Mateus': 24, 'Fernanda': 22, 'Tamires': 26, 'Cristiano': 25, 'Pedro': 23}
```

Dicionários

```
In [12]: estudantes_dict.clear()

In [13]: estudantes_dict
Out[13]: {}
```

```
In [14]: del estudantes_dict

In [15]: estudantes_dict
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-15-29b87ec1321c> in <module>()
----> 1 estudantes_dict

NameError: name 'estudantes_dict' is not defined
```

Dicionários - Outras operações

```
In [16]: estudantes = {"Mateus":24, "Fernanda":22, "Tamires":26, "Cristiano":25}

In [17]: estudantes
Out[17]: {'Mateus': 24, 'Fernanda': 22, 'Tamires': 26, 'Cristiano': 25}

In [18]: from operator import itemgetter
         sorted(estudantes.items())
Out[18]: [('Cristiano', 25), ('Fernanda', 22), ('Mateus', 24), ('Tamires', 26)]

In [19]: len(estudantes)
Out[19]: 4

In [20]: estudantes.keys()
Out[20]: dict_keys(['Mateus', 'Fernanda', 'Tamires', 'Cristiano'])

In [21]: estudantes.values()
Out[21]: dict_values([24, 22, 26, 25])
```

Dicionários - Outras operações

```
In [16]: estudantes = {"Mateus":24, "Fernanda":22, "Tamires":26, "Cristiano":25}

In [17]: estudantes
Out[17]: {'Mateus': 24, 'Fernanda': 22, 'Tamires': 26, 'Cristiano': 25}

In [18]: from operator import itemgetter
         sorted(estudantes.items())
Out[18]: [('Cristiano', 25), ('Fernanda', 22), ('Mateus', 24), ('Tamires', 26)]

In [19]: len(estudantes)
Out[19]: 4

In [20]: estudantes.keys()
Out[20]: dict_keys(['Mateus', 'Fernanda', 'Tamires', 'Cristiano'])

In [21]: estudantes.values()
Out[21]: dict_values([24, 22, 26, 25])
```

Dicionários - Outras operações

```
In [23]: estudantes2 = {"Maria":27, "Erika":28, "Milton":26}

In [24]: estudantes2
Out[24]: {'Maria': 27, 'Erika': 28, 'Milton': 26}

In [25]: estudantes.update(estudantes2)

In [26]: estudantes
Out[26]: {'Mateus': 24,
          'Fernanda': 22,
          'Tamires': 26,
          'Cristiano': 25,
          'Maria': 27,
          'Erika': 28,
          'Milton': 26}
```

Condicionais IF / Else

In [1]:
```
# Condicional If
if 5 > 2:
    print("Python funciona!")
```

```
Python funciona!
```

In [2]:
```
# Statement If...Else
if 5 < 2:
   print("Python funciona!")
else:
   print("Algo está errado!")
```

```
Algo está errado!
```

In [17]:
```
if dia == "Segunda":
  print("Hoje fará sol!")
elif dia == "Terça":
  print("Hoje vai chover!")
else:
  print("Sem previsão do tempo para o dia selecionado")
```

```
Hoje vai chover!
```

Condicionais IF / Else

In [15]:
```
# Usando mais de uma condição na cláusula if

disciplina = input('Digite o nome da disciplina: ')
nota_final = input('Digite a nota final (entre 0 e 100): ')

if disciplina == 'Geografia' and nota_final >= '70':
    print('Você foi aprovado!')
else:
    print('Lamento, acho que você precisa estudar mais!')
```

```
Digite o nome da disciplina: Geografia
Digite a nota final (entre 0 e 100): 50
Lamento, acho que você precisa estudar mais!
```

Condicionais IF / Else

In [16]:
```
# Usando mais de uma condição na cláusula if e introduzindo Placeholders

disciplina = input('Digite o nome da disciplina: ')
nota_final = input('Digite a nota final (entre 0 e 100): ')
semestre = input('Digite o semestre (1 a 4): ')

if disciplina == 'Geografia' and nota_final >= '50' and int(semestre) != 1:
    print('Você foi aprovado em %s com média final %r!' %(disciplina, nota_final))
else:
    print('Lamento, acho que você precisa estudar mais!')
```

```
Digite o nome da disciplina: Geografia
Digite a nota final (entre 0 e 100): 60
Digite o semestre (1 a 4): 4
Você foi aprovado em Geografia com média final '60'!
```

Condicionais IF - Aninhados

In [1]:
```
num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

```
Enter a number: 0
Zero
```

Laço de Repetição - For

```
# Criando uma lista e imprimindo cada um dos valores
ListaDoMercado = ["Leite", "Frutas", "Carne"]
for i in ListaDoMercado:
    print(i)
```

```
Leite
Frutas
Carne
```

In [3]:
```
for contador in range(0,5):
    print(contador)
```

```
0
1
2
3
4
```

Laço de Repetição - For

```
]: # Imprimindo na tela os números pares da lista de números
lista = [1,2,3,4,5,6,7,8,9,10]
for num in lista:
    if num % 2 == 0:
        print (num)
```

```
2
4
6
8
10
```

```
In [5]: for i in range(0,20,2):
            print(i)
```

```
0
2
4
6
8
10
12
14
16
18
```

Laço de Repetição - For

```
for caracter in 'Python':
    print (caracter)
```

```
P
y
t
h
o
n
```

```
# Pesquisando em listas
listaC = [5, 6, 7, 10, 50]

# Loop através da lista
for item in listaC:
    if item == 5:
        print("Número encontrado na lista!")
```

```
Número encontrado na lista!
```

```
for verificador in "Python":
    if verificador == "h":
        continue
    print(verificador)
```

```
P
y
t
o
n
```

Laço de Repetição - For

```
:  lista = ['Morango', 'Banana', 'Abacaxi', 'Uva']
   lista_tamanho = len(lista)
   for i in range(0, lista_tamanho):
       print(lista[i])
```

```
Morango
Banana
Abacaxi
Uva
```

Laço de Repetição - While

```
# Usando o loop while para imprimir os valores de 0 a 9
counter = 0
while counter < 10:
    print(counter)
    counter = counter + 1
```

```
0
1
2
3
4
5
6
7
8
9
```

:

```
counter = 0
while counter < 100:
    if counter == 4:
        break
    else:
        pass
    print(counter)
    counter = counter + 1
```

```
0
1
2
3
```

Laço de Repetição - While

```
: for i in range(2,30):
      j = 2
      counter = 0
      while j < i:
          if i % j == 0:
              counter = 1
              j = j + 1
          else:
              j = j + 1

      if counter == 0:
          print(str(i) + " é um número primo")
          counter = 0
      else:
          counter = 0
```

```
2 é um número primo
3 é um número primo
5 é um número primo
7 é um número primo
11 é um número primo
13 é um número primo
17 é um número primo
19 é um número primo
23 é um número primo
29 é um número primo
```

Função

In [1]:
```
# Definindo uma função
def primeiraFunc():
    print('Hello World')
```

In [2]:
```
primeiraFunc()
```

```
Hello World
```

In [3]:
```
# Definindo uma função com parâmetro
def primeiraFunc(nome):
    print('Hello %s' %(nome))
```

In [4]:
```
primeiraFunc('Aluno')
```

```
Hello Aluno
```

Função

In [5]:
```
def funcLeitura():
    for i in range(0, 5):
        print("Número " + str(i))
```

In [6]:
```
funcLeitura()
```

```
Número 0
Número 1
Número 2
Número 3
Número 4
```

```
def listaString(texto):
    print(texto.upper())
    return

listaString('Rumo à aprovação em concurso público')
```

```
RUMO À APROVAÇÃO EM CONCURSO PÚBLICO
```

In [7]:
```
# Função para somar números
def addNum(firstnum, secondnum):
    print("Primeiro número: " + str(firstnum))
    print("Segundo número: " + str(secondnum))
    print("Soma: ", firstnum + secondnum)
```

In [8]:
```
# Chamando a função e passando parâmetros
addNum(45, 3)
```

```
Primeiro número: 45
Segundo número: 3
Soma:  48
```

Função

In [9]:
```
import math

def numPrimo(num):
    '''
    Verificando se um número
    é primo.
    '''
    if (num % 2) == 0 and num > 2:
        return "Este número não é primo"
    for i in range(3, int(math.sqrt(num)) + 1, 2):
        if (num % i) == 0:
            return "Este número não é primo"
    return "Este número é primo"
```

In [10]:
```
numPrimo(541)
```

Out[10]: 'Este número é primo'

:
```
def listaPar():
    for i in range(2, 21, 2):
        print(i)

listaPar()
```

```
2
4
6
8
10
12
14
16
18
20
```

In [36]:
```
# Fazendo split dos dados
def split_string(text):
    return text.split(" ")
```

In [37]:
```
texto = "Esta função será bastante útil para separar grandes volumes de dados."
```

In [38]:
```
# Isso divide a string em uma lista.
print(split_string(texto))
```

```
['Esta', 'função', 'será', 'bastante', 'útil', 'para', 'separar', 'grandes', 'volumes', 'de', 'dados.']
```

Função - Lambda

```
In [3]: # Definindo uma expressão lambda
        potencia = lambda num: num**2
```

```
In [4]: potencia(5)
```

```
Out[4]: 25
```

```
In [5]: # Lembre: operadores de comparação retornam boolean, true or false
        Par = lambda x: x%2==0
```

```
In [6]: Par(3)
```

```
Out[6]: False
```

Função - Lambda

```
In [7]: first = lambda s: s[0]

In [8]: first('Python')
Out[8]: 'P'
```

```
In [9]: reverso = lambda s: s[::-1]

In [10]: reverso('Python')
Out[10]: 'nohtyP'
```

Módulos

In [2]:
```
# Importando apenas um dos métodos do módulo math
from math import sqrt
```

In [1]:
```
# Importando o módulo path
import math
```

In [6]:
```
# Imprimindo todos os métodos do módulo math
print(dir(math))
```

```
['__doc__', '__file__', '__loader__', '__name__', '__package__', '__spec__', 'acos', 'acosh', 'asin', 'asinh', 'atan', 'atan2'
'atanh', 'ceil', 'copysign', 'cos', 'cosh', 'degrees', 'e', 'erf', 'erfc', 'exp', 'expm1', 'fabs', 'factorial', 'floor', 'fmo
d', 'frexp', 'fsum', 'gamma', 'gcd', 'hypot', 'inf', 'isclose', 'isfinite', 'isinf', 'isnan', 'ldexp', 'lgamma', 'log', 'log1
0', 'log1p', 'log2', 'modf', 'nan', 'pi', 'pow', 'radians', 'sin', 'sinh', 'sqrt', 'tan', 'tanh', 'tau', 'trunc']
```

Módulos

```
In [3]: import random

In [4]: random.choice(['Maça', 'Banana', 'Laranja'])
Out[4]: 'Maça'

In [5]: from random import choice

In [7]: choice(['Maça', 'Banana', 'Laranja'])
Out[7]: 'Banana'
```

Q1) [INSTITUTO AOCP MJSP 2021] Uma cientista de dados necessita gerar um número randômico entre 1 e 15 na linguagem Python. Assinale a alternativa que apresenta o código correto que essa cientista de dados deve escrever.

a) init{

from random import randint print (randint(1, 15))}

b)from random import randint

print (randint(1, 15))

c)import random;

print (randint(1, 15));

d)import random print (random(1, 15))

e)from random import randnum print (randnum(1, 15))

Q1) [INSTITUTO AOCP MJSP 2021] Uma cientista de dados necessita gerar um número randômico entre 1 e 15 na linguagem Python. Assinale a alternativa que apresenta o código correto que essa cientista de dados deve escrever.

a) init{

from random import randint print (randint(1, 15))}

b)from random import randint

print(randint(1, 15))

c)import random;

print(randint(1, 15));

d)import random print (random(1, 15))

e)from random import randnum print (randnum(1, 15))

Q2) [INSTITUTO AOCP MJSP 2021] Um desenvolvedor implementou um programa para exibir a média de um dado retirado de uma grande base de dados. Para isso, foi utilizada a linguagem Python. O trecho do código que mostra o resultado é apresentado a seguir. Assinale a alternativa correta acerca desse trecho de código sabendo que a média do usuário foi 75.

```
print('Sua média foi {}.'.format(med))
```

a) O programa imprime: Sua média foi 75.

b) O programa imprime: Sua média foi {}.

c) O programa imprime: Sua média foi Null.

d) O programa apresenta um erro na impressão porque tenta converter tipo numérico em caractere.

e) O programa apresenta um erro na impressão, pois não apresenta o formato do valor.

Q2) [INSTITUTO AOCP MJSP 2021] Um desenvolvedor implementou um programa para exibir a média de um dado retirado de uma grande base de dados. Para isso, foi utilizada a linguagem Python. O trecho do código que mostra o resultado é apresentado a seguir. Assinale a alternativa correta acerca desse trecho de código sabendo que a média do usuário foi 75.

```
print('Sua média foi {}.'.format(med))
```

a) O programa imprime: Sua média foi 75.

b) O programa imprime: Sua média foi {}.

c) O programa imprime: Sua média foi Null.

d) O programa apresenta um erro na impressão porque tenta converter tipo numérico em caractere.

e) O programa apresenta um erro na impressão, pois não apresenta o formato do valor.

Q3) [INSTITUTO AOCP MJSP 2021] Analise o seguinte código Python, escrito por um cientista de dados, e assinale a alternativa correta.

In [01]: num = 88 In [02]: print ('par' if num%2 == 0 else 'ímpar')

a) O resultado da execução de código é a informação ‘ímpar’ na tela.

b) O código apresenta como resultado “1” na linha [02], pois num%2 == 0 é uma atribuição e não uma comparação.

c) O código executado apresenta na tela o resultado ‘0’.

d) O resultado da execução de código é a informação ‘par’ na tela.

e) O código gera um erro, pois na linha [02] o comando if...else não está correto.

Q3) [INSTITUTO AOCP MJSP 2021] Analise o seguinte código Python, escrito por um cientista de dados, e assinale a alternativa correta.

In [01]: num = 88 In [02]: print ('par' if num%2 == 0 else 'ímpar')

a) O resultado da execução de código é a informação ‘ímpar’ na tela.

b) O código apresenta como resultado “1” na linha [02], pois num%2 == 0 é uma atribuição e não uma comparação.

c) O código executado apresenta na tela o resultado ‘0’.

d) O resultado da execução de código é a informação ‘par’ na tela.

e) O código gera um erro, pois na linha [02] o comando if...else não está correto.

Q4) [INSTITUTO AOCP MJSP 2021] Analise o seguinte código Python, escrito por um cientista de dados, e assinale a alternativa correta.

In [01]: nome = input("Qual o seu nome? ") In[02]: print(nome)

a) O código não irá funcionar pois está faltando ‘;’ ao final de cada linha.

b) O uso da função input está incorreto, pois ela tem que ser chamada em uma nova linha.

c) O resultado do comando print é apresentar na tela o nome informado pelo usuário.

d) Falta a estrutura correta de criação de uma classe para a execução do comando.

e) Falta definir o tipo da variável nome, por isso o programa não irá executar.

Q4) [INSTITUTO AOCP MJSP 2021] Analise o seguinte código Python, escrito por um cientista de dados, e assinale a alternativa correta.

In [01]: nome = input("Qual o seu nome? ") In[02]: print(nome)

a) O código não irá funcionar pois está faltando ‘;’ ao final de cada linha.

b) O uso da função input está incorreto, pois ela tem que ser chamada em uma nova linha.

c) O resultado do comando print é apresentar na tela o nome informado pelo usuário.

d) Falta a estrutura correta de criação de uma classe para a execução do comando.

e) Falta definir o tipo da variável nome, por isso o programa não irá executar.

Q5) [INSTITUTO AOCP MJSP 2020] Um analista do MJSP armazenou nomes e ID no seguinte programa Python:

```
01 dados = {
02 'Machado de Assis': 7575,
03 'Santos Dumont': 7475,
04 'Rui Barbosa': 1675,
05 };
```

Agora, o analista necessita apresentar em tela o ID 7475 referente ao nome informado. Assinale a alternativa que apresenta o código correto para imprimir esse ID na tela para o usuário.

a) print(7475);

b) print(dados[nome], ID==7475);

c) print(ID, ‘Santos Dumont’);

d) print(dados == ‘Santos Dumont);

e) print(dados['Santos Dumont']);

Q5) [INSTITUTO AOCP MJSP 2020] Um analista do MJSP armazenou nomes e ID no seguinte programa Python:

```
01 dados = {
02 'Machado de Assis': 7575,
03 'Santos Dumont': 7475,
04 'Rui Barbosa': 1675,
05 };
```

Agora, o analista necessita apresentar em tela o ID 7475 referente ao nome informado. Assinale a alternativa que apresenta o código correto para imprimir esse ID na tela para o usuário.

a) print(7475);

b) print(dados[nome], ID==7475);

c) print(ID, ‘Santos Dumont’);

d) print(dados == ‘Santos Dumont);

e) print(dados['Santos Dumont']);

Q6) [IBADE Prefeitura de Jaru 2020] Dos itens abaixo, aquele que é uma linguagem de programação:

a) Firefox.

b) Explorer.

c) Chrome.

d) Python.

e) Safari.

Q7) [QUADRIX CREA-TO 2019] Quanto aos conceitos e às técnicas de programação de computadores, julgue o item.

Em um programa escrito em linguagem Python, o comando de atribuição x = int(5.9) fará com que a variável x passe a armazenar um valor inteiro igual a 6.

Q6) [IBADE Prefeitura de Jaru 2020] Dos itens abaixo, aquele que é uma linguagem de programação:

a) Firefox.

b) Explorer.

c) Chrome.

d) Python.

e) Safari.

Q7) [QUADRIX CREA-TO 2019] Quanto aos conceitos e às técnicas de programação de computadores, julgue o item.

Em um programa escrito em linguagem Python, o comando de atribuição x = int(5.9) fará com que a variável x passe a armazenar um valor inteiro igual a 6. ERRADO.

Q8) [ADM&TEC Prefeitura de Olivença 2020] Leia as afirmativas a seguir:

I. Com o PHP, é possível enviar e receber cookies, por exemplo.

II. O Python é uma linguagem multiplataforma que serve para desenvolver aplicações para celulares, desktop e web, por exemplo.

Marque a alternativa CORRETA:

a) As duas afirmativas são verdadeiras.

b) A afirmativa I é verdadeira, e a II é falsa.

c) A afirmativa II é verdadeira, e a I é falsa.

d) As duas afirmativas são falsas.

Q8) [ADM&TEC Prefeitura de Olivença 2020] Leia as afirmativas a seguir:

I. Com o PHP, é possível enviar e receber cookies, por exemplo.

II. O Python é uma linguagem multiplataforma que serve para desenvolver aplicações para celulares, desktop e web, por exemplo.

Marque a alternativa CORRETA:

a) As duas afirmativas são verdadeiras.

b) A afirmativa I é verdadeira, e a II é falsa.

c) A afirmativa II é verdadeira, e a I é falsa.

d) As duas afirmativas são falsas.

Q9) [UFCG UFCG 2019] Assinale a opção abaixo que contém somente informações corretas.

a) Python 3 possui retrocompatibilidade total com Python 2.

b) Python 3 não é compatível com cadeias de caracteres (strings) Unicode.

c) count(d) retorna o número de elementos do dict d.

d) Dicionários em Python 3.7 preservam a ordem de inserção.

e) Utiliza-se array.add(x) para adicionar x a array.

Q9) [UFCG UFCG 2019] Assinale a opção abaixo que contém somente informações corretas.

a) Python 3 possui retrocompatibilidade total com Python 2.

b) Python 3 não é compatível com cadeias de caracteres (strings) Unicode.

c) count(d) retorna o número de elementos do dict d.

d) Dicionários em Python 3.7 preservam a ordem de inserção.

e) Utiliza-se array.add(x) para adicionar x a array.

Obs: **Essa implementação ocorreu a partir da versão 3.6 do Python. Antes era necessário usar OrderedDict para ordenar por inserção**.

Q10) [QUADRIX Prefeitura de Jataí 2019] Na linguagem de programação Python, o método que permite verificar se uma string começa com alguns caracteres é o

a) startswith.

b) return.

c) def

d) len.

e) count.

Q10) [QUADRIX Prefeitura de Jataí 2019] Na linguagem de programação Python, o método que permite verificar se uma string começa com alguns caracteres é o

a) startswith.

b) return.

c) def

d) len.

e) count.

Q11) [COVEST UFPE 2019] Considere uma String s, que armazena o valor "ALO MUNDO”. Utilizando Java e Python, respectivamente, a alternativa com as instruções que exibiriam a substring “MU” seria:

a) Java: s.substring(4,6)

Python: s.substring(5,7)

b) Java: s[5,6]

Python: s[5:6]

c) Java: s.substring(4,6)

Python: s[-5:-3]

d) Java: s.substring(-4,-3)

Python: s[5:6].

e) Java: s.substring(4,6)

Python: s.substr(4,6)

Q11) [COVEST UFPE 2019] Considere uma String s, que armazena o valor "ALO MUNDO”. Utilizando Java e Python, respectivamente, a alternativa com as instruções que exibiriam a substring “MU” seria:

a) Java: s.substring(4,6)

Python: s.substring(5,7)

b) Java: s[5,6]

Python: s[5:6]

c) Java: s.substring(4,6)

Python: s[-5:-3]

d) Java: s.substring(-4,-3)

Python: s[5:6].

e) Java: s.substring(4,6)

Python: s.substr(4,6)

Q12) [FCC TJMA 2019] Para que o programa seja executado corretamente, em condições ideais, a indicação I deve ser substituída por:

```
I
.....
  for x in pro:
    print(x)
processos = ["0456789908", "0875643087", "0897645109"]
exibir_processos(processos)
```

Para que o programa seja executado corretamente, em condições ideais, a indicação I deve ser substituída por:

a) private exibir_processos(pro):

b) public exibir_processos(pro):

c) function exibir_processos(pro):

d) definition exibir_processos(pro):

e) def exibir_processos(pro):

Q12) [FCC TJMA 2019] Para que o programa seja executado corretamente, em condições ideais, a indicação I deve ser substituída por:

```
I
.....
  for x in pro:
    print(x)
processos = ["0456789908", "0875643087", "0897645109"]
exibir_processos(processos)
```

Para que o programa seja executado corretamente, em condições ideais, a indicação I deve ser substituída por:

a) private exibir_processos(pro):

b) public exibir_processos(pro):

c) function exibir_processos(pro):

d) definition exibir_processos(pro):

e) def exibir_processos(pro):