Algoritmos e Estruturas de Dados I

LISTAS

Prof. Tiago Eugenio de Melo
tmelo@uea.edu.br

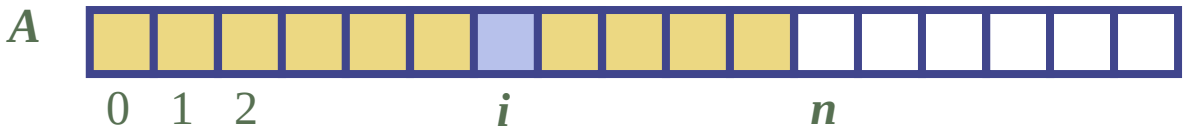
www.tiagodemelo.info

Observações

- O conteúdo dessa aula é parcialmente proveniente do Capítulo 5 do livro “*Data Structures and Algorithms in Python*”.
- As palavras com a fonte `Courier` indicam uma palavra-reservada da linguagem de programação.

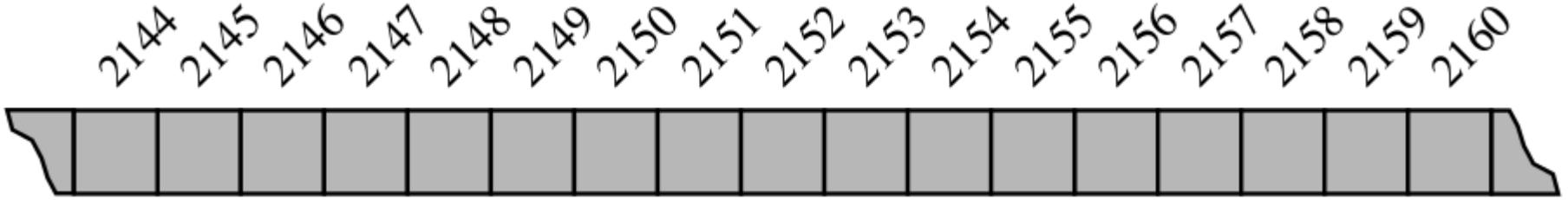
Introdução

- Python tem os tipos primitivos **list**, **tuple** e **str**.
- Cada um desses tipos é uma sequência que suporta indexar o acesso a cada elemento de uma sequência, usando uma sintaxe tal como A[i].
- Cada um desses tipos usa um array para representar a sequência:
  - Um array (~vetor) é um conjunto de memória alocada que pode ser acessada usando índices consecutivos.



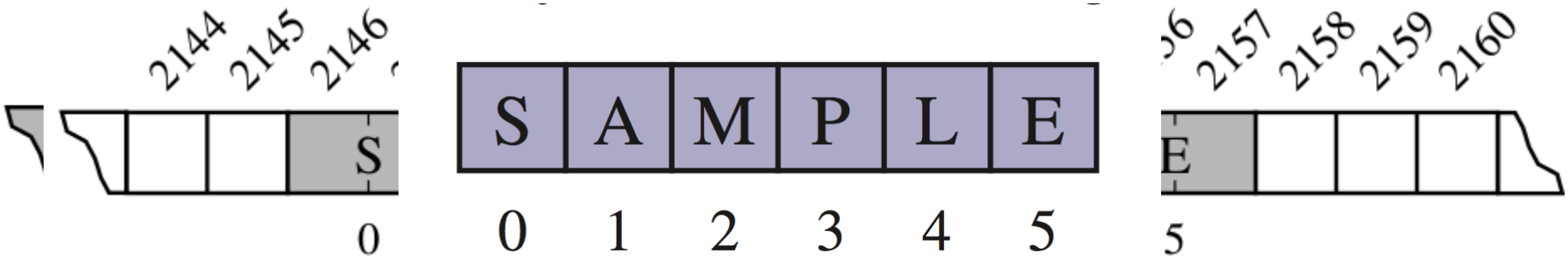
Arrays

- Podemos considerar um array como um grupo de variáveis que podem ser armazenadas em uma porção contínua de memória.



Arrays

- Exemplo:
  - Python representa um caracter com 16 bits (2 bytes).
  - Portanto, a string 'SAMPLE' seria representada assim:



Array (referências)

Array (referências)

- Considere que queremos armazenar uma lista de nomes em Python.

Array (referências)

- Considere que queremos armazenar uma lista de nomes em Python.
- Exemplo:

Array (referências)

- Considere que queremos armazenar uma lista de nomes em Python.
- Exemplo:

```
['Rene', 'Joseph', 'Janet', 'Jonas', 'Helen', 'Virginia', ... ]
```

Array (referências)

- Considere que queremos armazenar uma lista de nomes em Python.
- Exemplo:

```
['Rene', 'Joseph', 'Janet', 'Jonas', 'Helen', 'Virginia', ... ]
```

- Cada elemento é uma string, mas naturalmente cada string terá um tamanho diferente.

Array (referências)

- Considere que queremos armazenar uma lista de nomes em Python.
- Exemplo:

```
['Rene', 'Joseph', 'Janet', 'Jonas', 'Helen', 'Virginia', ... ]
```

- Cada elemento é uma string, mas naturalmente cada string terá um tamanho diferente.
- Python representa uma lista ou tupla usando um mecanismo de um array de referências a objetos.

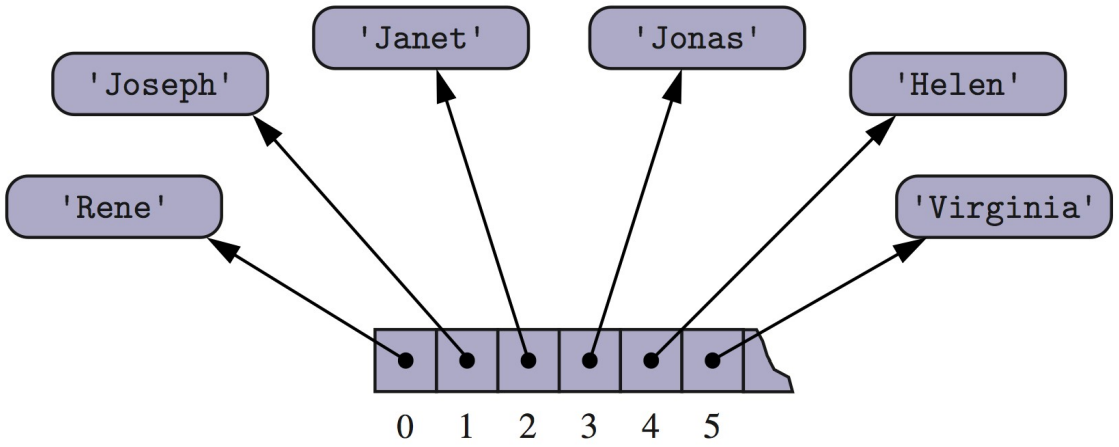
Array (referências)

Array (referências)

- Exemplo:

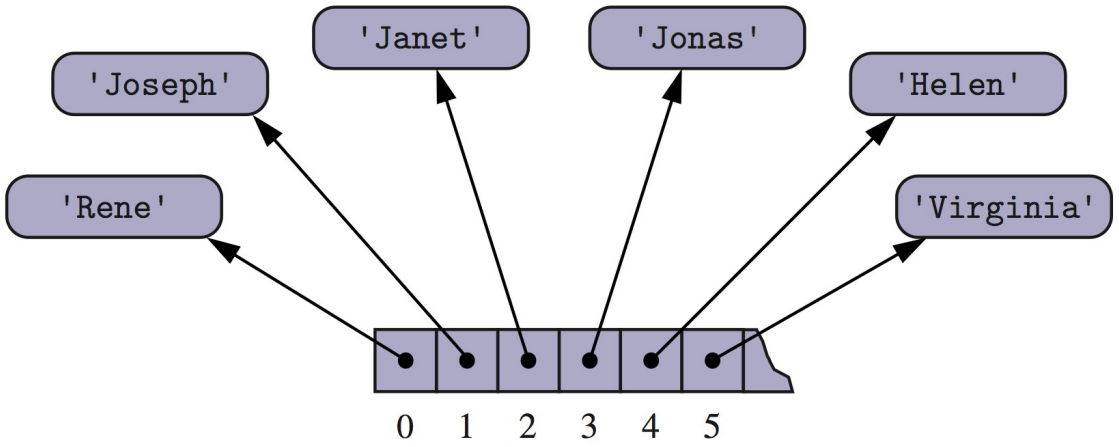
Array (referências)

- Exemplo:



Array (referências)

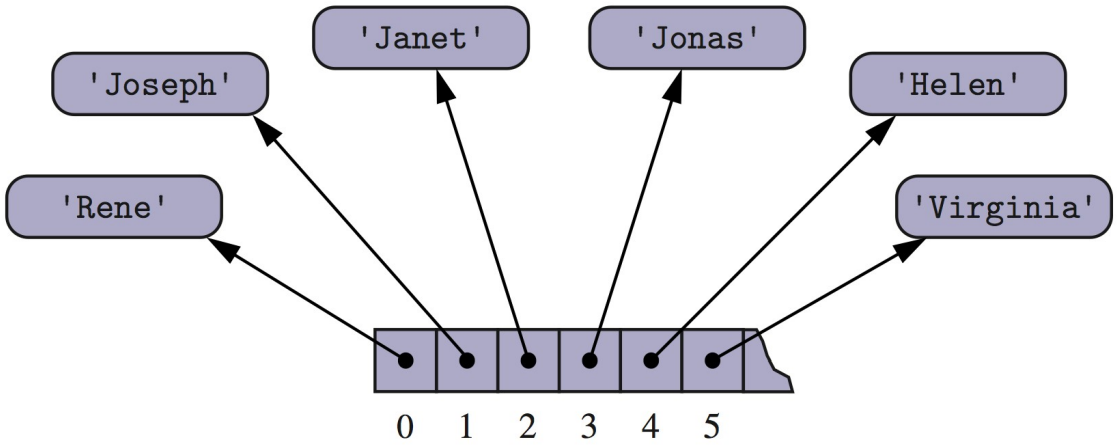
- Exemplo:



- Embora os tamanhos dos nomes possam variar, a quantidade de bits usados para armazenar o endereço de cada elemento é fixo.

Array (referências)

- Exemplo:



- Embora os tamanhos dos nomes possam variar, a quantidade de bits usados para armazenar o endereço de cada elemento é fixo.
- Assim, Python pode realizar o tempo constante de acesso aos elementos de uma lista e tupla baseado no seu índice.

Array (referências)

- Considere:

Array (referências)

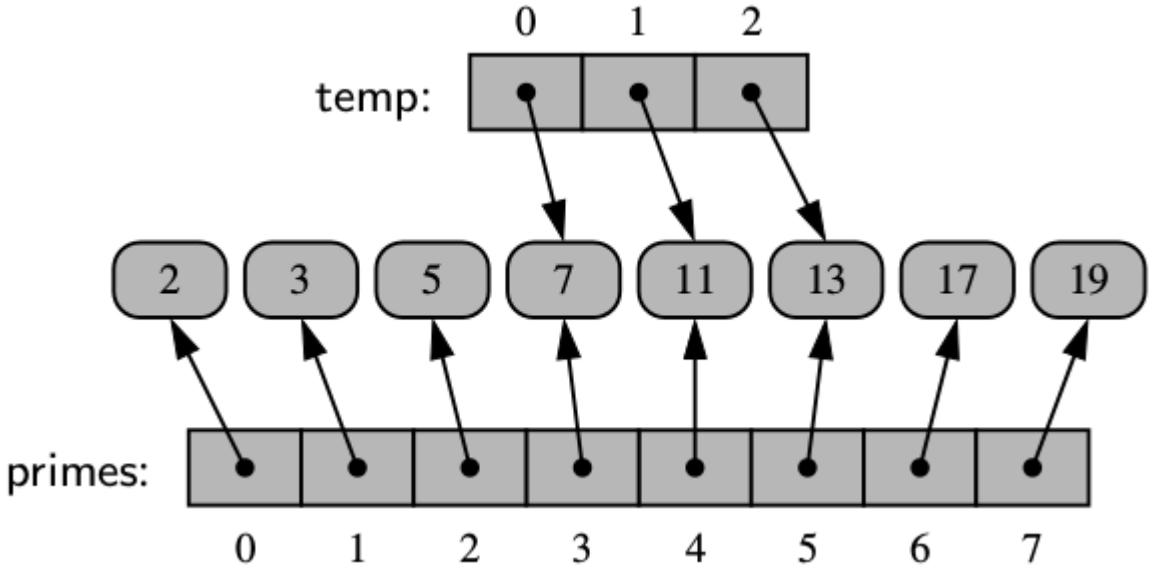
- Considere:
  - `primes = [2, 3, 5, 7, 11, 13, 17, 19]`

Array (referências)

- Considere:
  - `primes = [2, 3, 5, 7, 11, 13, 17, 19]`
  - `temp = primes[3:6]`

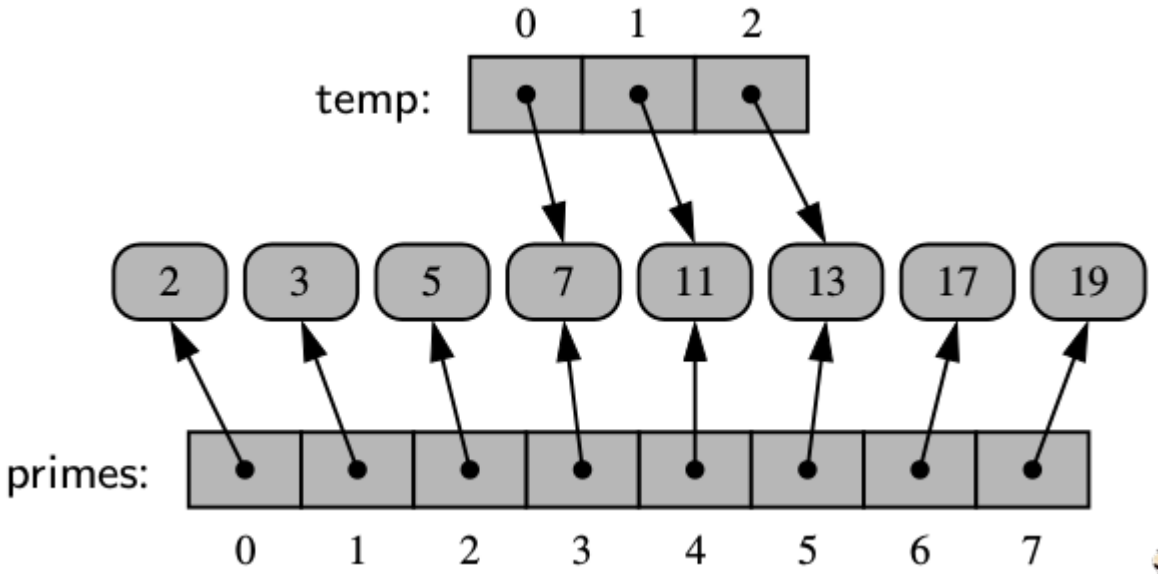
Array (referências)

- Considere:
  - `primes = [2, 3, 5, 7, 11, 13, 17, 19]`
  - `temp = primes[3:6]`



Array (referências)

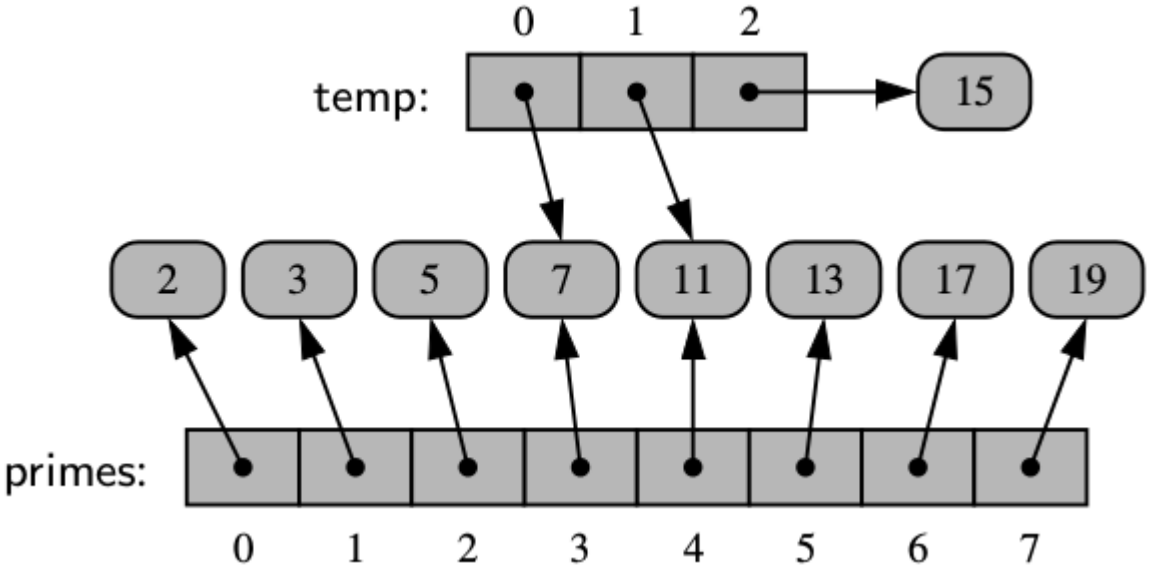
- Considere:
  - `primes = [2, 3, 5, 7, 11, 13, 17, 19]`
  - `temp = primes[3:6]`



O que acontece em:
`temp[2] = 15`

Array (referências)

- Considere:
  - `primes = [2, 3, 5, 7, 11, 13, 17, 19]`
  - `temp = primes[3:6]`



O que acontece em:
`temp[2] = 15`

Array (referências)

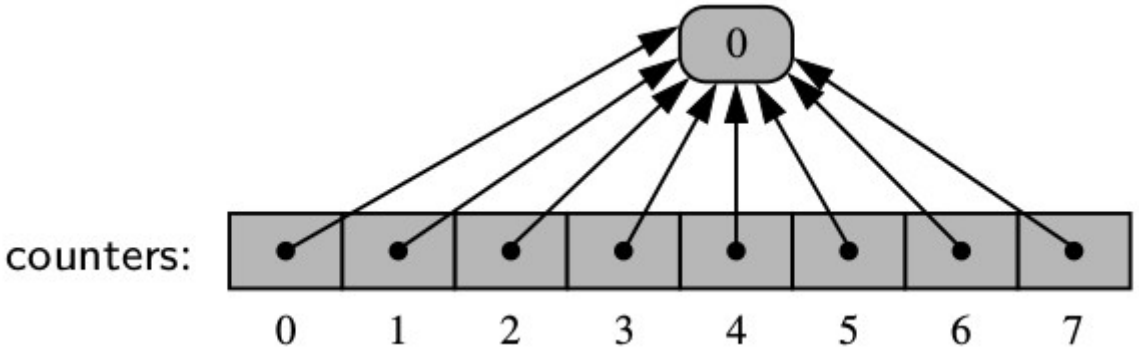
- Considere:

Array (referências)

- Considere:
  - `counters = [0] * 8`

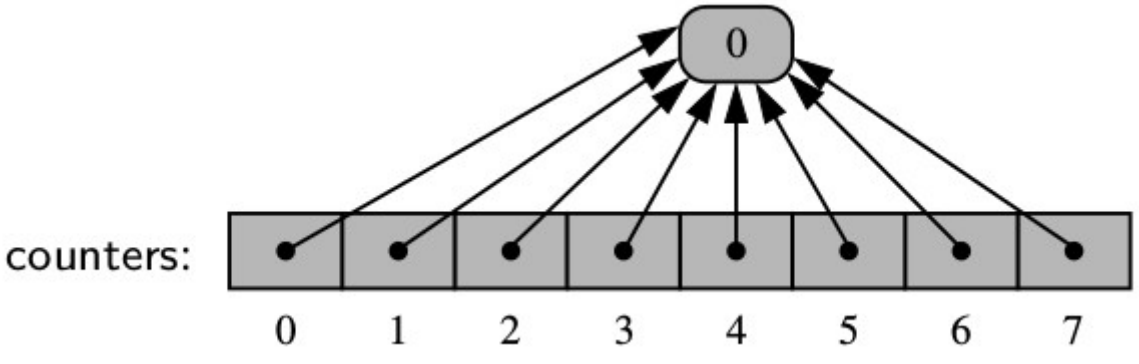
Array (referências)

- Considere:
  - `counters = [0] * 8`



Array (referências)

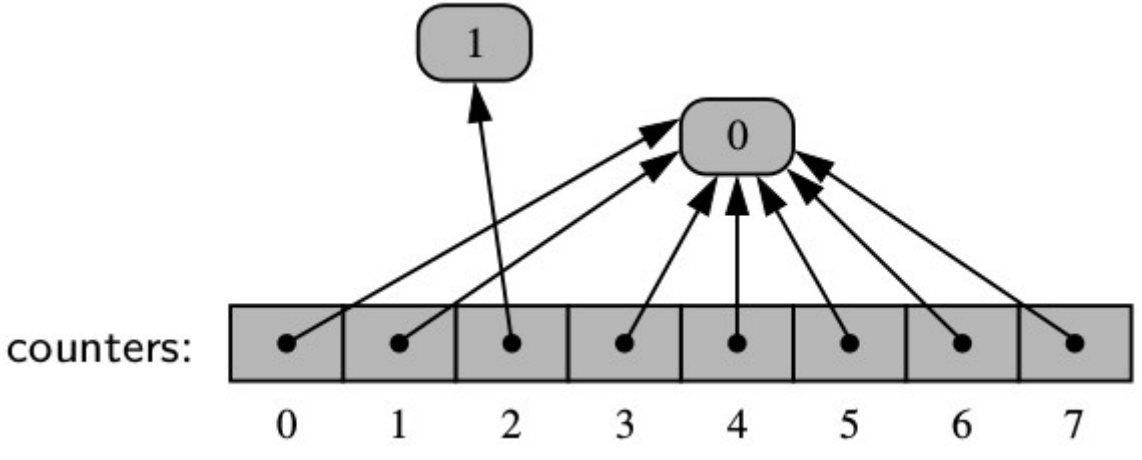
- Considere:
  - `counters = [0] * 8`



O que acontece em:
`counters[2] += 1`

Array (referências)

- Considere:
  - `counters = [0] * 8`



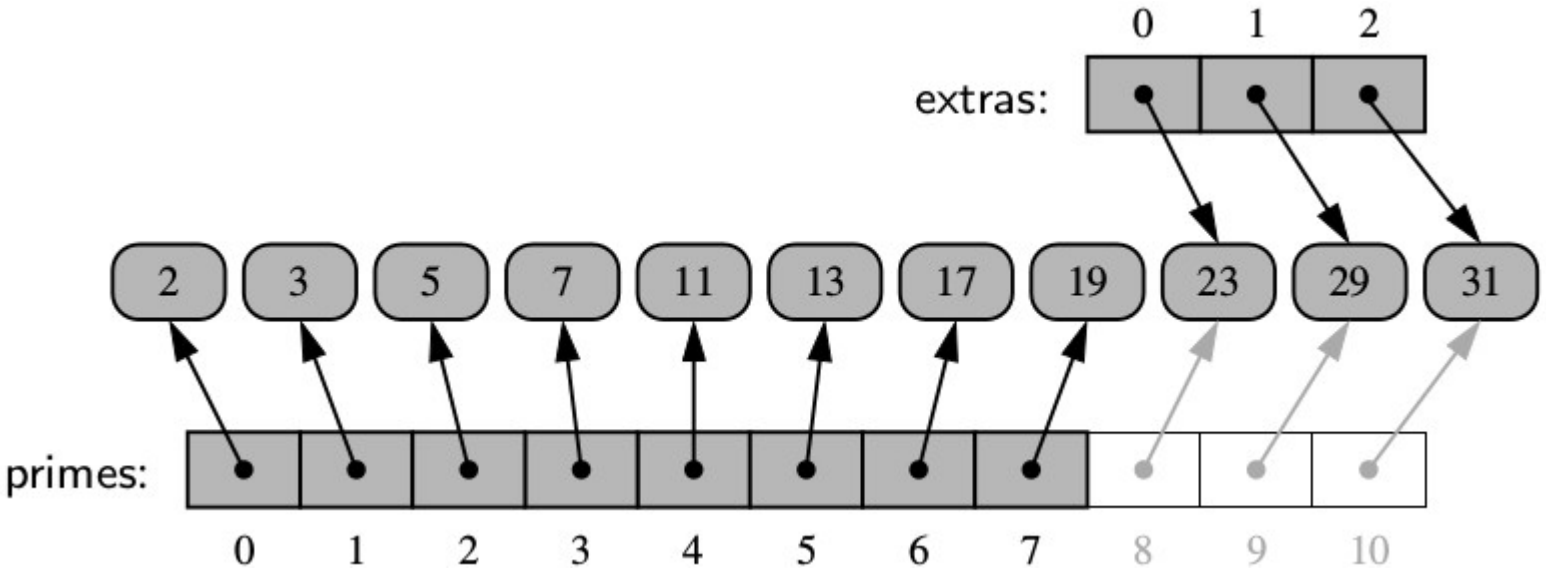
O que acontece em:
`counters[2] += 1`

Array (referências)

- E no caso de `extend`?

Array (referências)

- E no caso de `extend`?



Compact Arrays

Compact Arrays

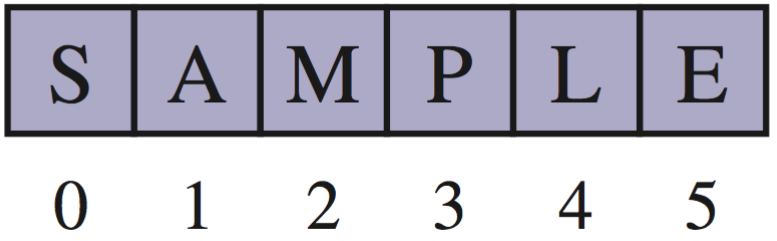
- Strings em Python são representadas como um array de caracteres (e **não** um array de referências).

Compact Arrays

- Strings em Python são representadas como um array de caracteres (e **não** um array de referências).
- Esse tipo de representação é também chamada de *compact array* porque o array está armazenando os bits que representam os dados primários (caracteres, no caso das strings).

Compact Arrays

- Strings em Python são representadas como um array de caracteres (e **não** um array de referências).
- Esse tipo de representação é também chamada de *compact array* porque o array está armazenando os bits que representam os dados primários (caracteres, no caso das strings).



Compact Arrays

- Compact arrays têm algumas vantagens sobre estruturas referenciais em termos de performance computacional:

Compact Arrays

- Compact arrays têm algumas vantagens sobre estruturas referenciais em termos de performance computacional:
  - Em geral, a quantidade de memória é menor.

Compact Arrays

- Compact arrays têm algumas vantagens sobre estruturas referenciais em termos de performance computacional:
  - Em geral, a quantidade de memória é menor.
  - Os dados primários são armazenados consecutivamente na memória.

Compact Arrays

Compact Arrays

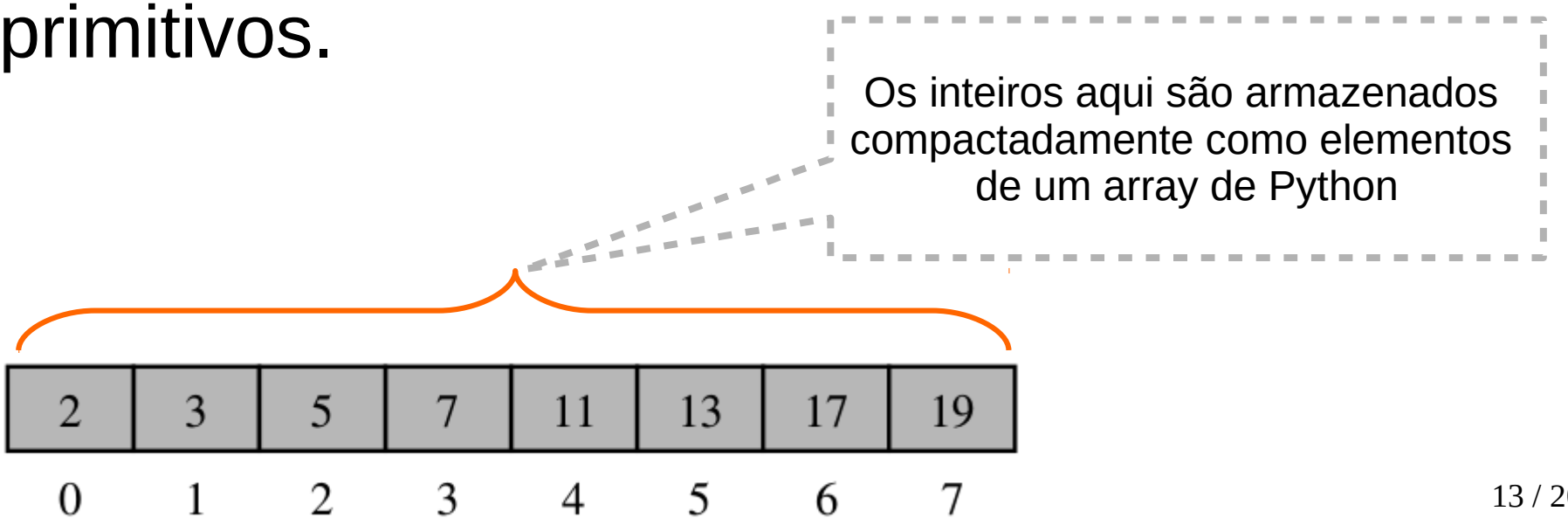
- Esse recurso está disponível em um módulo chamado **array**.

Compact Arrays

- Esse recurso está disponível em um módulo chamado **array**.
- Este módulo define uma classe, também chamada de array, fornecendo armazenamento compact para arrays de tipos de dados primitivos.

Compact Arrays

- Esse recurso está disponível em um módulo chamado **array**.
- Este módulo define uma classe, também chamada de array, fornecendo armazenamento compact para arrays de tipos de dados primitivos.



Compact Arrays

- Exemplo:

Compact Arrays

- Exemplo:

```
from array import array
primos = array('i', [2, 3, 5, 7, 11, 13, 19])
```

Compact Arrays

- Exemplo:

```
from array import array
primos = array('i', [2, 3, 5, 7, 11, 13, 19])
```

```
>>> type (primos)
<type 'array.array'>
```

Compact Arrays

- Exemplo:

```
from array import array
primos = array('i', [2, 3, 5, 7, 11, 13, 19])
```

```
>>> type (primos)
<type 'array.array'>
```

```
>>> primos.append(10)
>>> primos
array('i', [2, 3, 5, 7, 23, 13, 19, 10])
```

Compact Arrays

- Exemplo:

```
from array import array
primos = array('i', [2, 3, 5, 7, 11, 13, 19])
```

```
>>> type (primos)
<type 'array.array'>
```

```
>>> primos.append(10)
>>> primos
array('i', [2, 3, 5, 7, 23, 13, 19, 10])
```

```
>>> primos[4] = 23
>>> primos
array('i', [2, 3, 5, 7, 23, 13, 19])
```

Compact Arrays

- Exemplo:

```
from array import array
primos = array('i', [2, 3, 5, 7, 11, 13, 19])
```

```
>>> type (primos)
<type 'array.array'>
```

```
>>> primos.append(10)
>>> primos
array('i', [2, 3, 5, 7, 23, 13, 19, 10])
```

```
>>> primos[4] = 23
>>> primos
array('i', [2, 3, 5, 7, 23, 13, 19])
```

```
>>> primos[3] = 2.3

Traceback (most recent call last):
  File "<pyshell#22>", line 1, in <module>
    primos[3] = 2.3
TypeError: integer argument expected, got float
```

Compact Arrays

- Tipos de códigos suportados pelo módulo array:

Code	C Data Type	Typical Number of Bytes
`'b'`	signed char	1
`'B'`	unsigned char	1
`'u'`	Unicode char	2 or 4
`'h'`	signed short int	2
`'H'`	unsigned short int	2
`'i'`	signed int	2 or 4
`'I'`	unsigned int	2 or 4
`'l'`	signed long int	4
`'L'`	unsigned long int	4
`'f'`	float	4
`'d'`	float	8

Array Dinâmico

Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.

Array Dinâmico

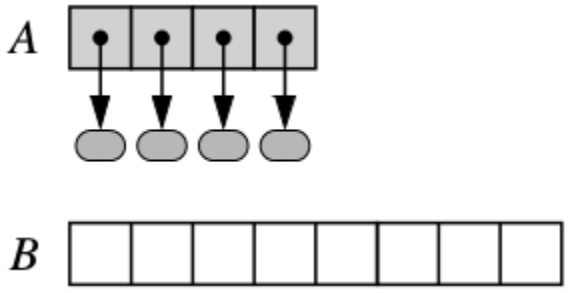
- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:

Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).

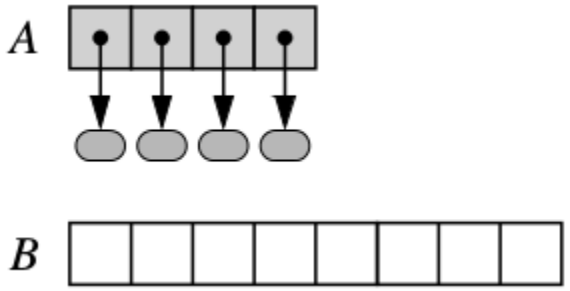
Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).



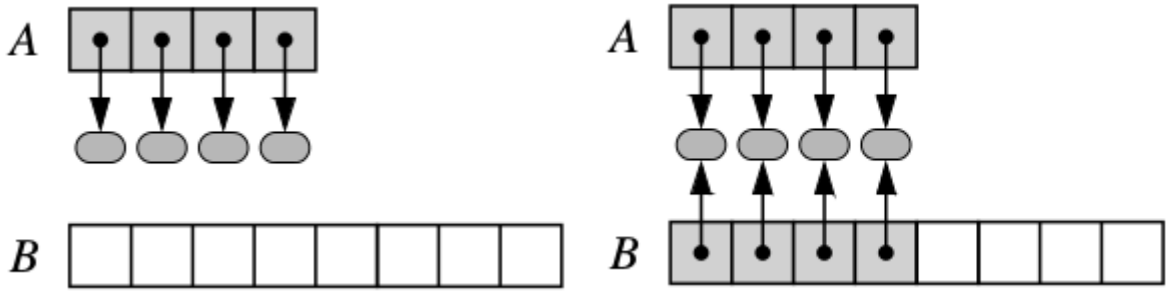
Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).
  - Armazena os elementos de A em B.



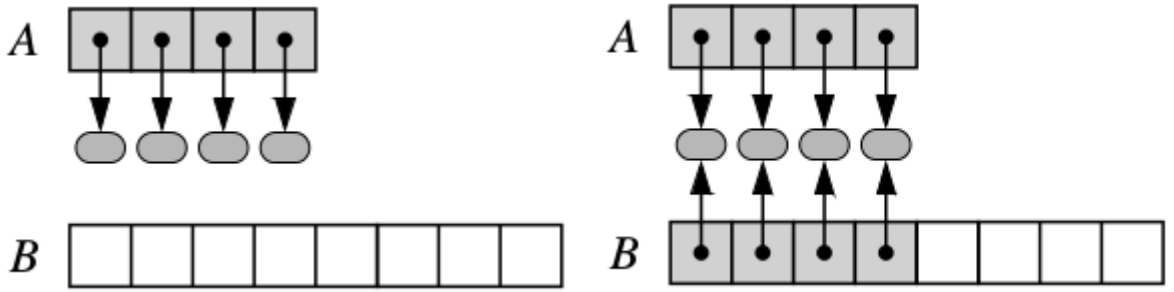
Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).
  - Armazena os elementos de A em B.



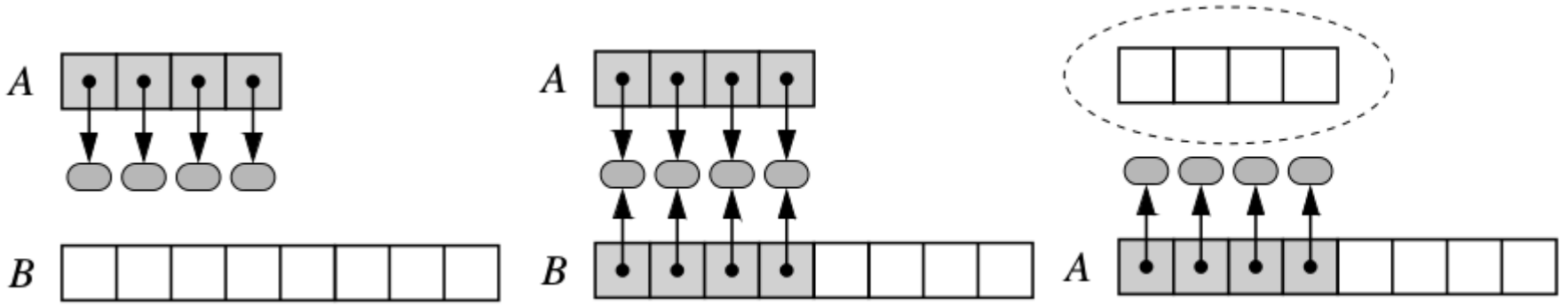
Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).
  - Armazena os elementos de A em B.
  - Muda as referências de A.



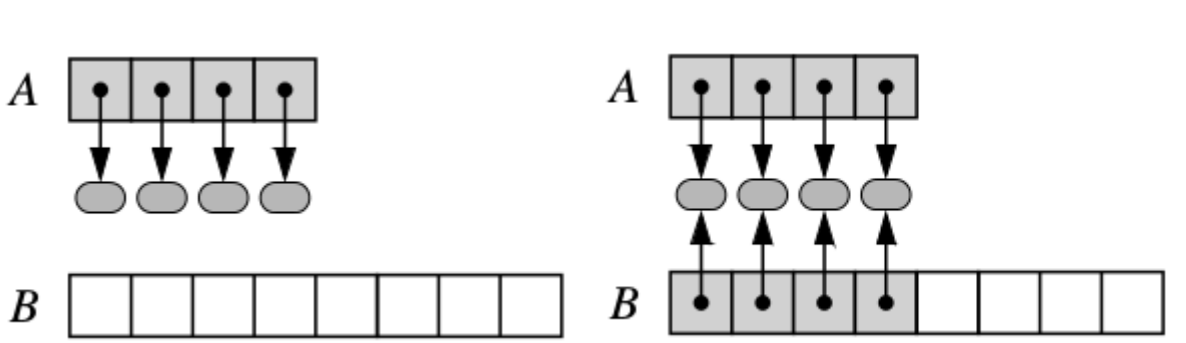
Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).
  - Armazena os elementos de A em B.
  - Muda as referências de A.



Array Dinâmico

- Python possui uma classe altamente otimizada para arrays dinâmicos.
- Porém, a ideia geral é fazer isso nos seguintes passos:
  - Aloca um novo array B (maior que A).
  - Armazena os elementos de A em B.
  - Muda as referências de A.



Exemplo Prático

- Estrutura para armazenar jogadores com os maiores scores.

Exemplo Prático

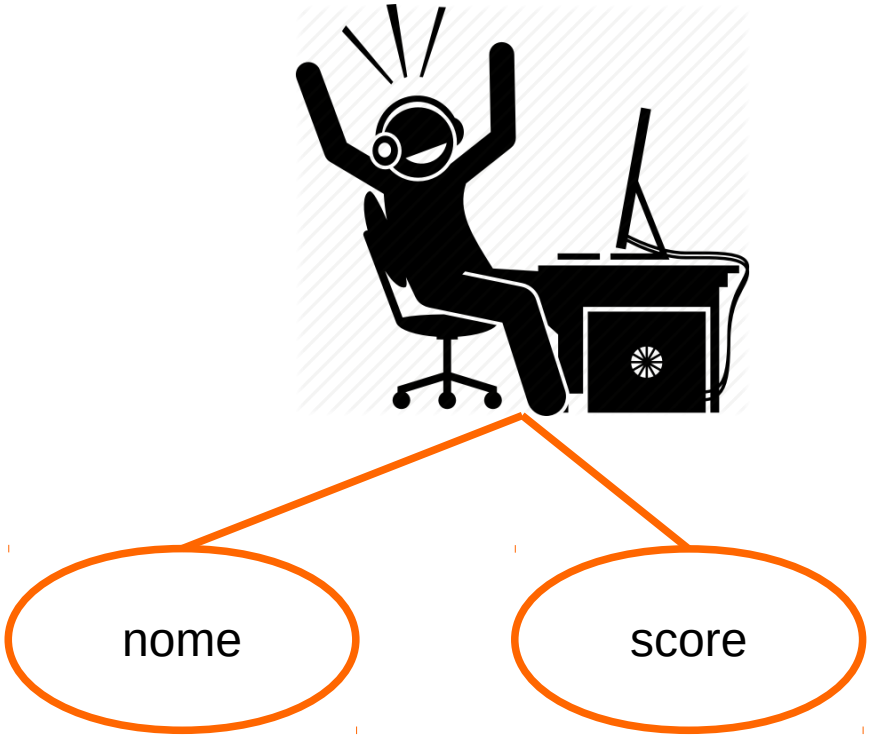
- Estrutura para armazenar jogadores com os maiores scores.

Exemplo Prático

- Estrutura para armazenar jogadores com os maiores scores.

Exemplo Prático

- Estrutura para armazenar jogadores com os maiores scores.



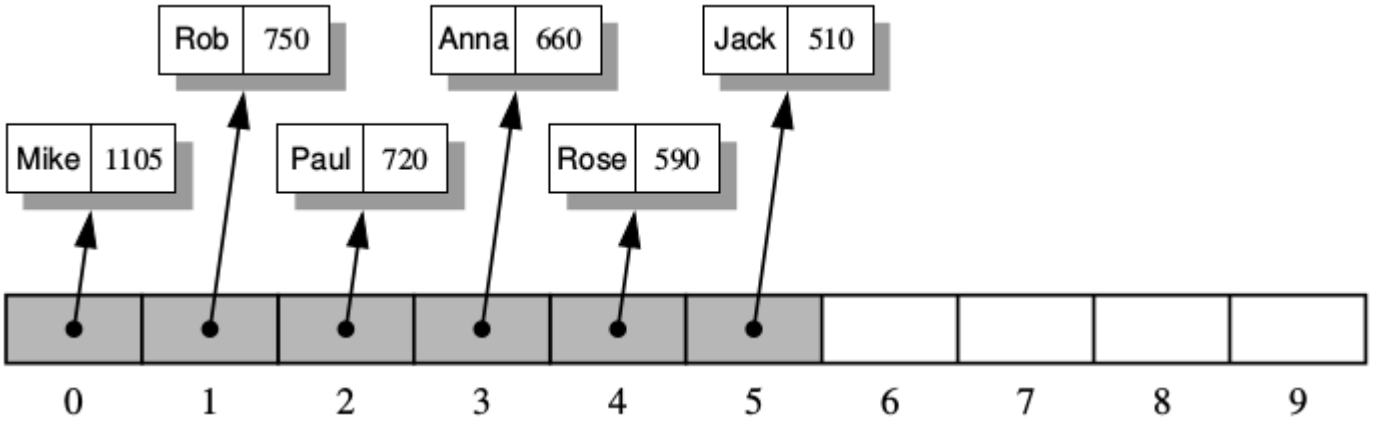
Exemplo Prático

Exemplo Prático

- Versão inicial

Exemplo Prático

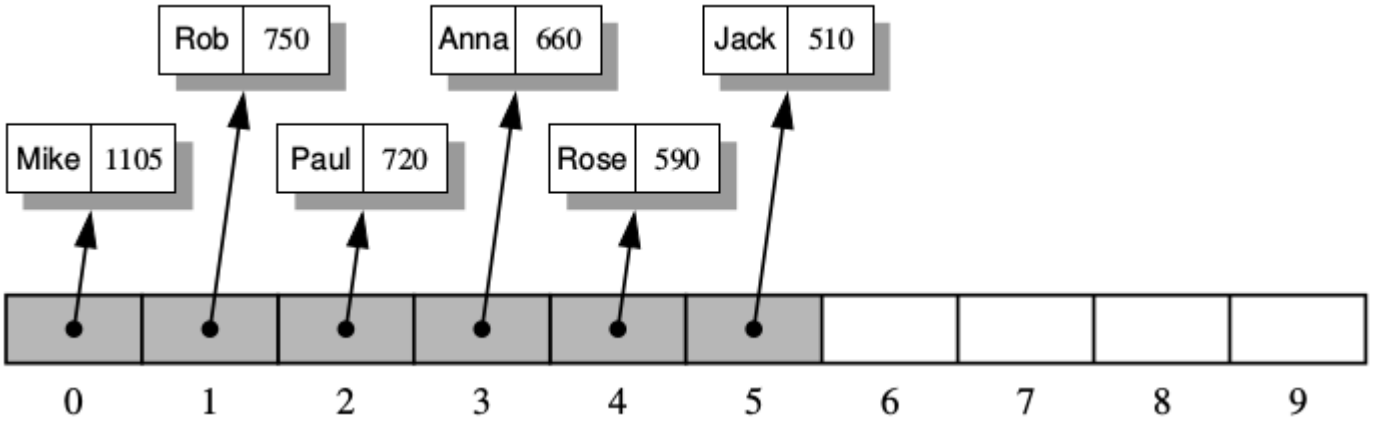
- Versão inicial



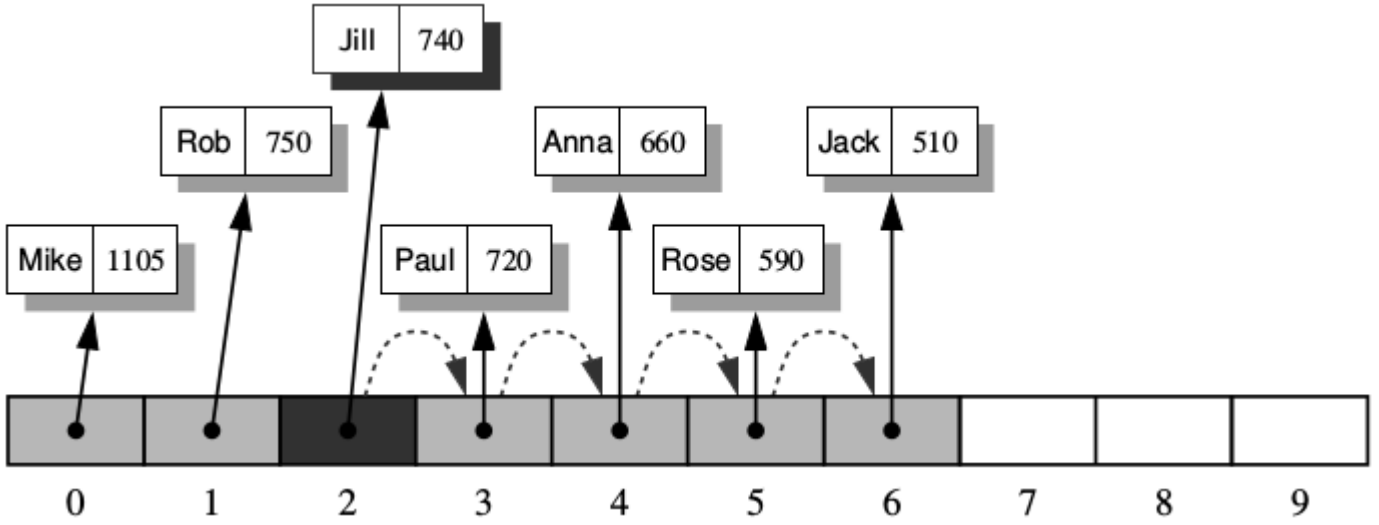
- Adicionando um novo elemento

Exemplo Prático

- Versão inicial



- Adicionando um novo elemento



Exemplo Prático

- Classe `GameEntry`

```
class GameEntry:
  """Represents one entry of a list of high scores."""
```

cria um objeto (name, score)
retorna o nome
retorna o score
retorna o par (nome, score)

Exemplo Prático

- Classe `GameEntry`

```
class GameEntry:
  """Represents one entry of a list of high scores."""

  def __init__(self, name, score):
    """Create an entry with given name and score."""
    self._name = name
    self._score = score
```

retorna o nome

retorna o score

retorna o par (nome, score)

Exemplo Prático

- Classe `GameEntry`

```
class GameEntry:
  """Represents one entry of a list of high scores."""

  def __init__(self, name, score):
    """Create an entry with given name and score."""
    self._name = name
    self._score = score

  def get_name(self):
    """Return the name of the person for this entry."""
    return self._name
```

retorna o score

retorna o par (nome, score)

Exemplo Prático

- Classe `GameEntry`

```
class GameEntry:
  """Represents one entry of a list of high scores."""

  def __init__(self, name, score):
    """Create an entry with given name and score."""
    self._name = name
    self._score = score

  def get_name(self):
    """Return the name of the person for this entry."""
    return self._name

  def get_score(self):
    """Return the score of this entry."""
    return self._score
```

retorna o par (nome, score)

Exemplo Prático

- Classe `GameEntry`

```
class GameEntry:
  """Represents one entry of a list of high scores."""

  def __init__(self, name, score):
    """Create an entry with given name and score."""
    self._name = name
    self._score = score

  def get_name(self):
    """Return the name of the person for this entry."""
    return self._name

  def get_score(self):
    """Return the score of this entry."""
    return self._score

  def __str__(self):
    """Return string representation of the entry."""
    return '({0}, {1})'.format(self._name, self._score) # e.g., '(Bob, 98)'
```

```
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""
```

método construtor

retorna uma entrada

retorna uma string com a representação do maior score

adiciona um objeto

```
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""

  def __init__(self, capacity=10):
    """Initialize scoreboard with given maximum capacity.

    All entries are initially None.
    """
    self._board = [None] * capacity        # reserve space for future scores
    self._n = 0                            # number of actual entries
```

retorna uma entrada

retorna uma string com a representação do maior score

adiciona um objeto

```
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""

  def __init__(self, capacity=10):
    """Initialize scoreboard with given maximum capacity.

    All entries are initially None.
    """
    self._board = [None] * capacity        # reserve space for future scores
    self._n = 0                            # number of actual entries

  def __getitem__(self, k):
    """Return entry at index k."""
    return self._board[k]
```

retorna uma string com a representação do maior score

adiciona um objeto

```
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""

  def __init__(self, capacity=10):
    """Initialize scoreboard with given maximum capacity.

    All entries are initially None.
    """
    self._board = [None] * capacity        # reserve space for future scores
    self._n = 0                            # number of actual entries

  def __getitem__(self, k):
    """Return entry at index k."""
    return self._board[k]

  def __str__(self):
    """Return string representation of the high score list."""
    return '\n'.join(str(self._board[j]) for j in range(self._n))
```

adiciona um objeto

```
class Scoreboard:
  """Fixed-length sequence of high scores in nondecreasing order."""

  def __init__(self, capacity=10):
    """Initialize scoreboard with given maximum capacity.

    All entries are initially None.
    """
    self._board = [None] * capacity        # reserve space for future scores
    self._n = 0                            # number of actual entries

  def __getitem__(self, k):
    """Return entry at index k."""
    return self._board[k]

  def __str__(self):
    """Return string representation of the high score list."""
    return '\n'.join(str(self._board[j]) for j in range(self._n))

  def add(self, entry):
    """Consider adding entry to high scores."""
    score = entry.get_score()

    # Does new entry qualify as a high score?
    # answer is yes if board not full or score is higher than last entry
    good = self._n < len(self._board) or score > self._board[-1].get_score()

    if good:
      if self._n < len(self._board):       # no score drops from list
        self._n += 1                       # so overall number increases

      # shift lower scores rightward to make room for new entry
      j = self._n - 1
      while j > 0 and self._board[j-1].get_score() < score:
        self._board[j] = self._board[j-1]  # shift entry from j-1 to j
        j -= 1                             # and decrement j
      self._board[j] = entry               # when done, add new entry
```