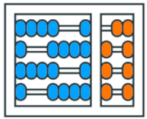


Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 - Aula 10

Sets e Diccionários

Algoritmos e Programação de Computadores

Turmas

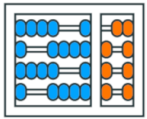
OVXZ

Prof. Lise R. R. Navarrete

lrommel@ic.unicamp.br

Terça-feira, 26 de abril de 2022

21:00h - 23:00h (CB06)



Instituto de Computação

UNIVERSIDADE ESTADUAL DE CAMPINAS

UNICAMP

MC102 – Algoritmos e Programação de Computadores

Turmas

OVXZ

https://ic.unicamp.br/~mc102/

Site da Coordenação de MC102

Aulas teóricas:

Terça-feira, 21:00h - 23:00h (CB06)

Quinta-feira, 19:00h - 21:00h (CB06)

Conteúdo

- Sets
- Dicionários
  - Exemplos usando Dicionários

Sets

Sets

- Sets são usados para armazenar vários items em uma simples variavel.
- Set é um dos 4 tipos internos em Python usados para armazenar coleções de dados. Os outros 3 são: Listas, Tuplas e Diccionarios, todos com diferente qualidade e uso.
- Set é uma coleção de dados que é não ordenada, imutável e não indexado.
- Os elementos de um Set não podem ser alterados, mas podemos remover elementos e adicionar novos.
- Elementos duplicados não são permitidos.
- Sets são escritos com chaves:

```
set1 = {"A","B","C"}

print(set1)

print(len(set1))
```

```
{'A', 'C', 'B'}
3
```

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length

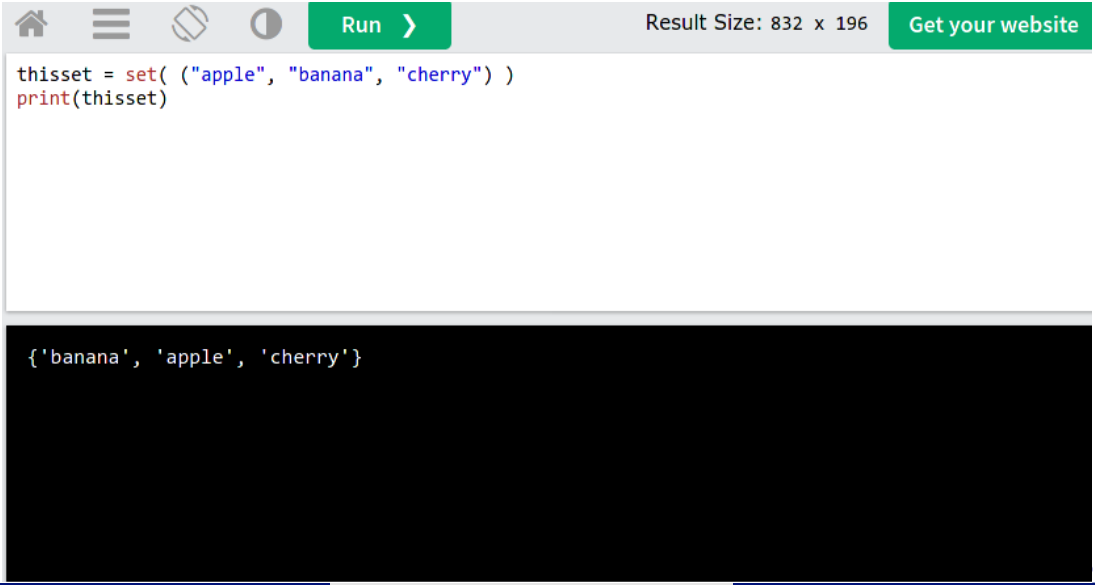



Run ❯ Result Size: 832 x 196 Get your website

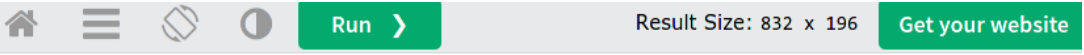
```
set1 = {"apple", "banana", "cherry"}
set2 = {1, 5, 7, 9, 3}
set3 = {True, False, False}
set4 = {"abc", 34, True, 40, "male"}

print(set4)
print(type(set4))
```

```
{True, 34, 40, 'male', 'abc'}
<class 'set'>
```

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



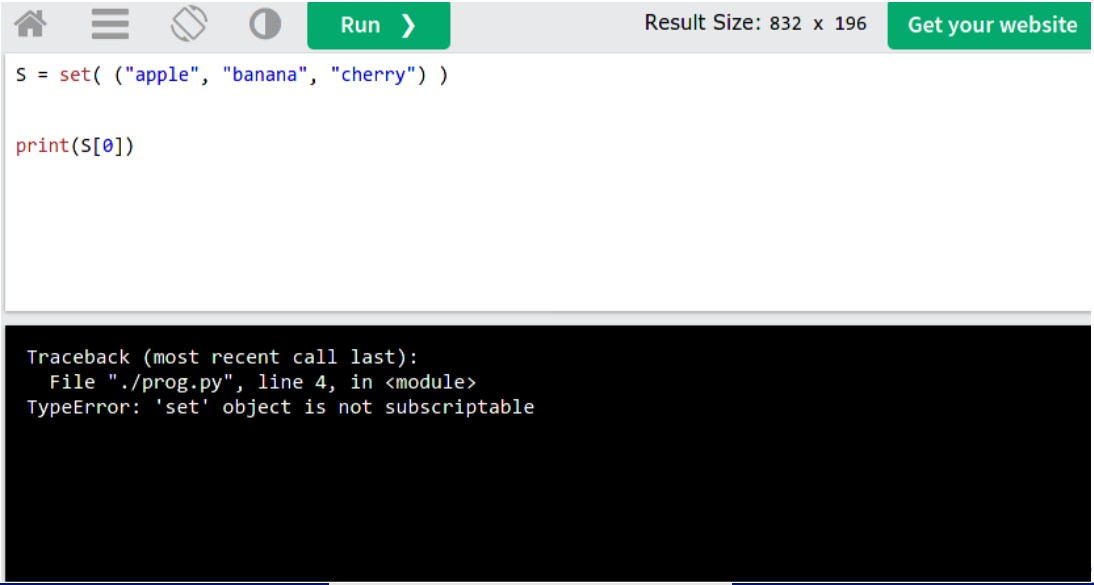
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



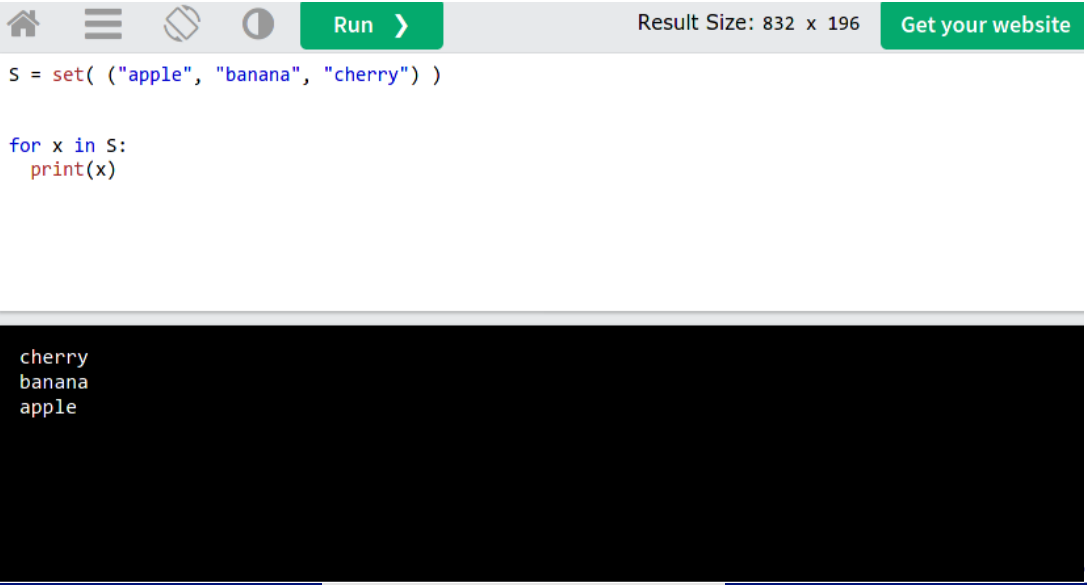

```
thisset = set( ("apple", "banana", "cherry") )
print(thisset)


thisset = set( ["apple", "banana", "cherry"] )
print(thisset)
```

```
{'apple', 'cherry', 'banana'}
{'apple', 'cherry', 'banana'}
```

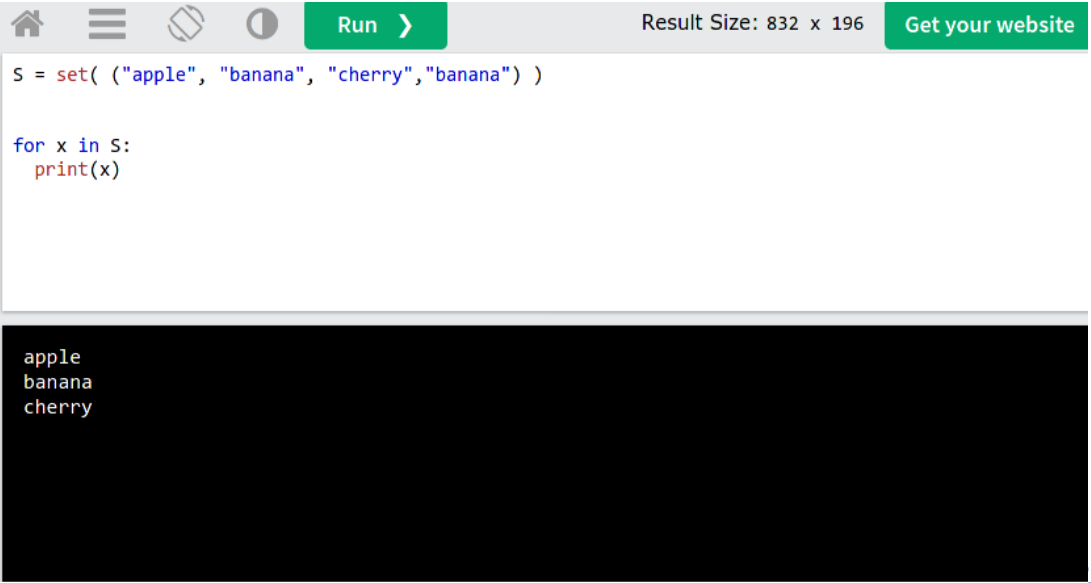
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



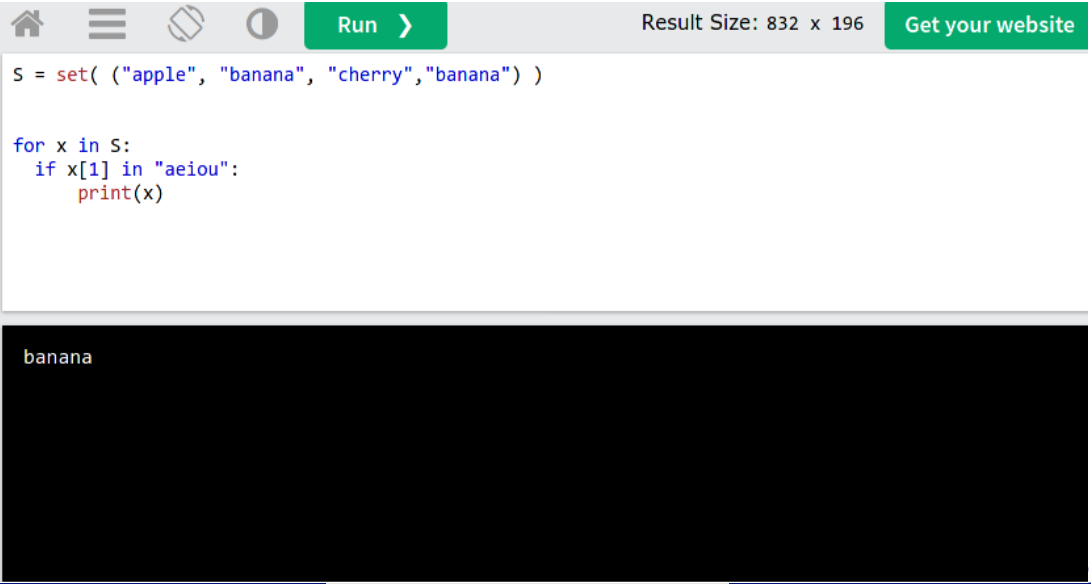
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



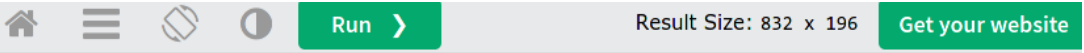
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



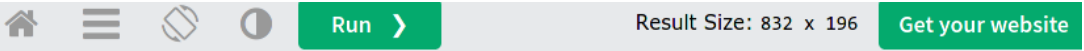
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length




```
S = set( ("apple", "banana", "cherry","banana") )

S.add("orange")
S.add("orange")
S.add("coconut")


for x in S:
  if x[1] in "aeiou":
      print(x)
```

```
coconut
banana
```

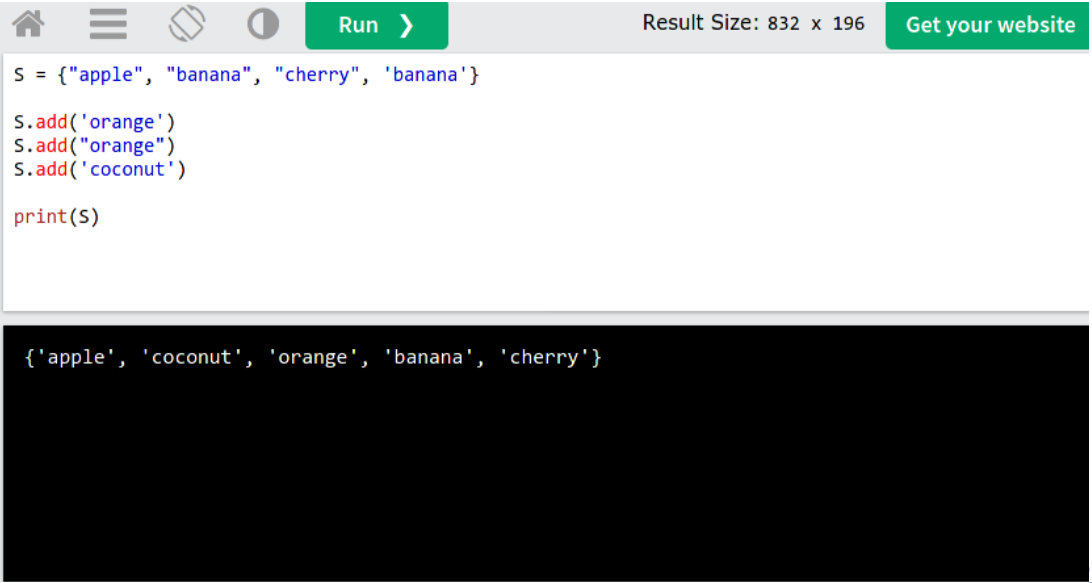
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



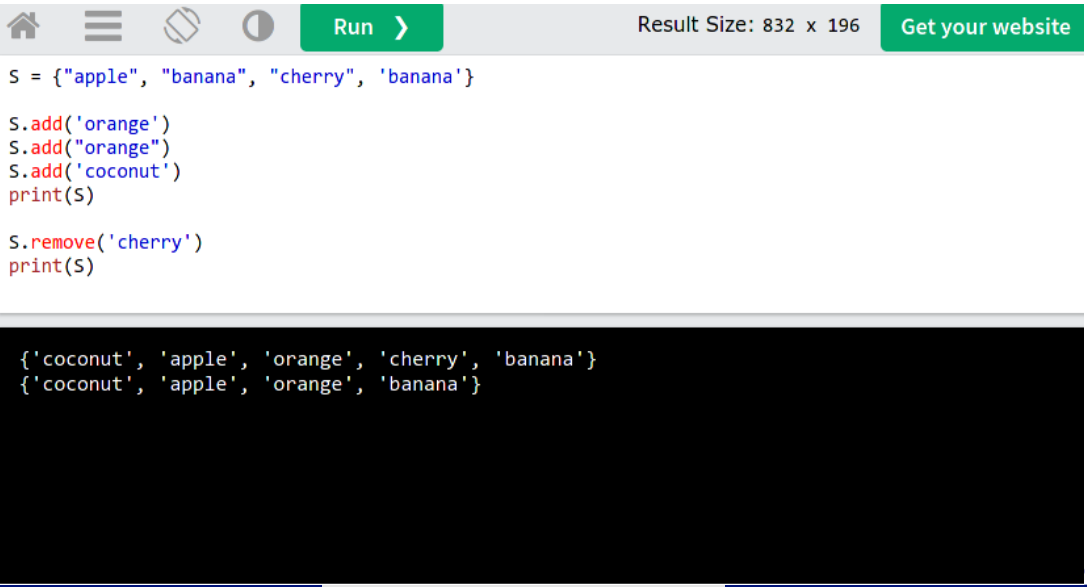

```
S = set( ("apple", "banana", "cherry","banana") )

S.add("orange")
S.add("orange")
S.add("coconut")

print(S)
for x in S:
  if x[1] in "aeiou":
      print(x)
```

```
{'cherry', 'banana', 'coconut', 'orange', 'apple'}
banana
coconut
```

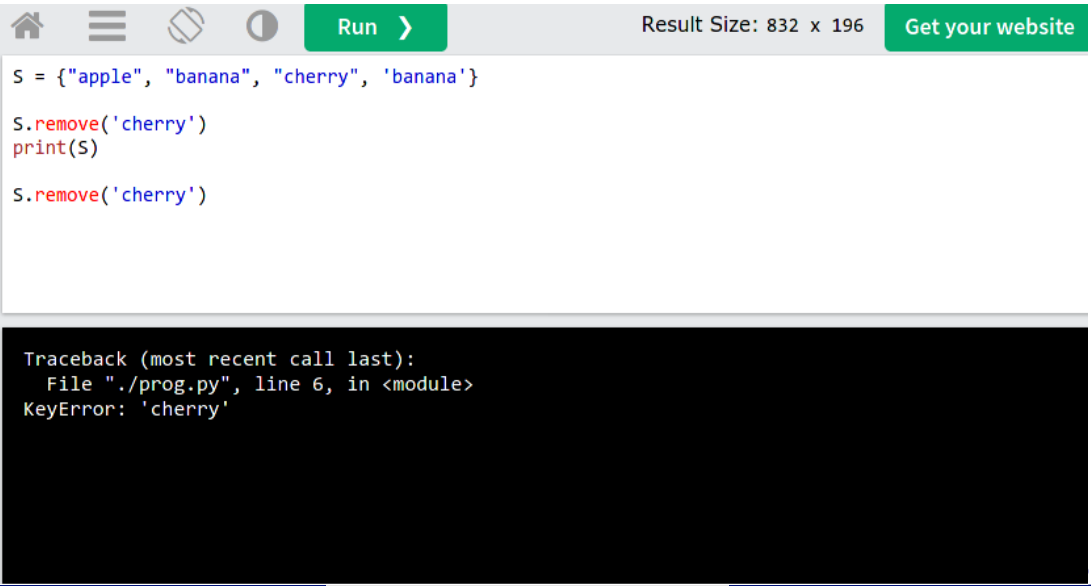
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



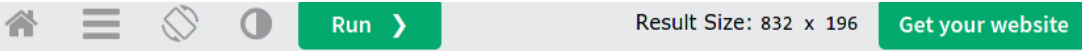
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length

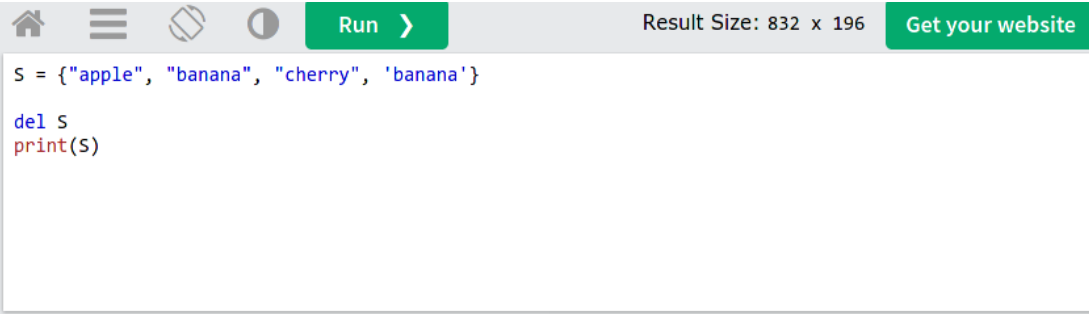
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



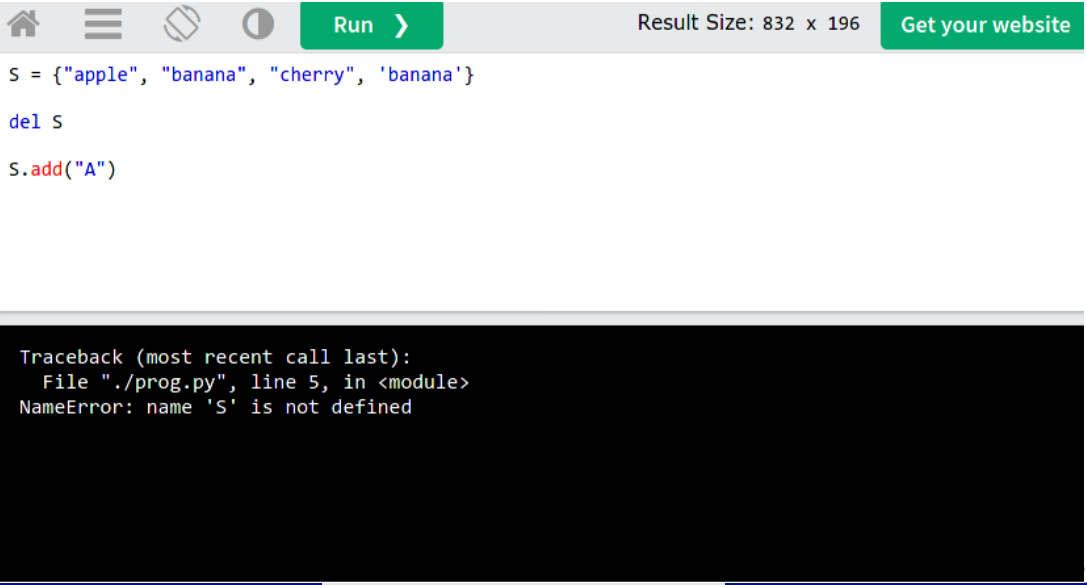

```
S = {"apple", "banana", "cherry", 'banana'}

S.clear()
print(S)

S.add(True)
print(S)
```

```
set()
{True}
```

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length

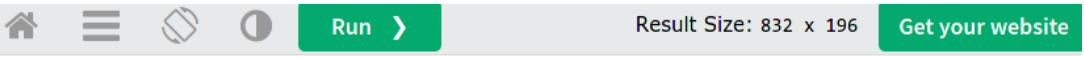



Run ❯ Result Size: 832 x 196 Get your website

```
S1 = {"apple", "banana", "cherry", 'banana'}
S2 = {"orange","orange"}
S3 = S1 + S2
print(S3)
```

```
Traceback (most recent call last):
  File "./prog.py", line 3, in <module>
TypeError: unsupported operand type(s) for +: 'set' and 'set'
```

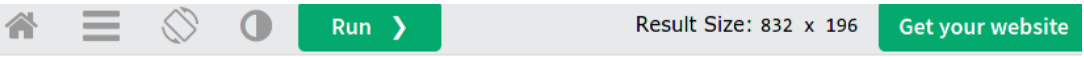
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length




```
S1 = {"apple", "banana", "cherry", 'banana'}
S2 = {"orange","orange"}

S3 = S1.union(S2)

print(S1,S2,S3,sep="\n")
```

```
{'banana', 'cherry', 'apple'}
{'orange'}
{'banana', 'orange', 'cherry', 'apple'}
```

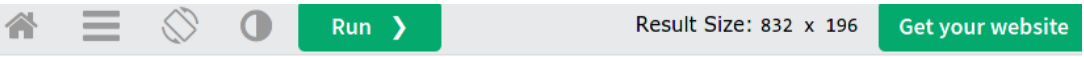
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length




```
S1 = {"apple", "banana", "cherry", 'banana'}
S2 = {"orange","orange"}

S3 = S1.union(S2)

print(S1,S2,S3,sep="\n")

S3.update("123456")
print(S3)
```

```
{'cherry', 'banana', 'apple'}
{'orange'}
{'cherry', 'orange', 'banana', 'apple'}
{'1', '4', '2', 'apple', '6', '3', 'cherry', 'orange', '5', 'banana'}
```

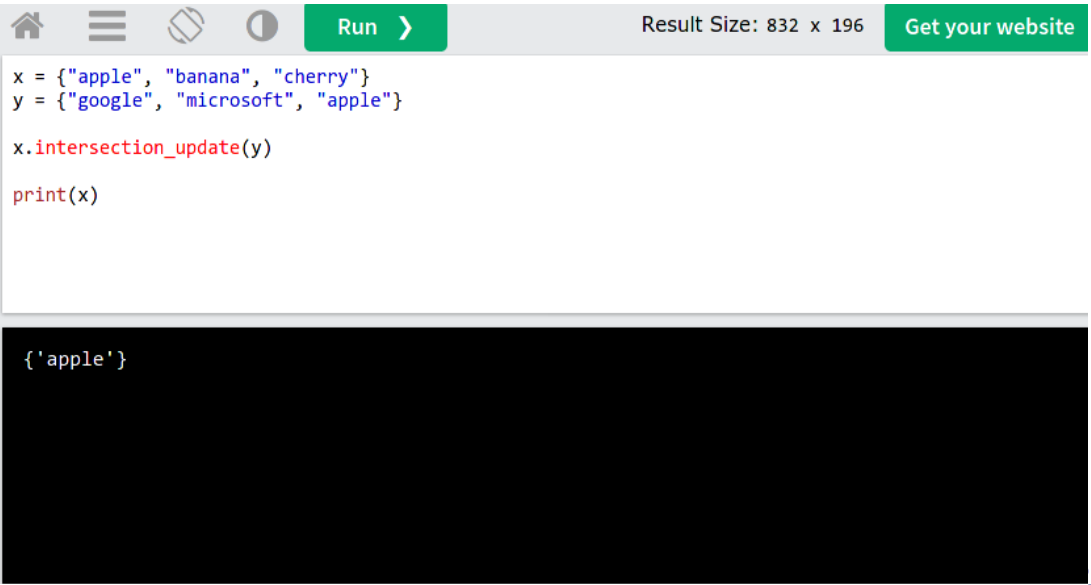
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



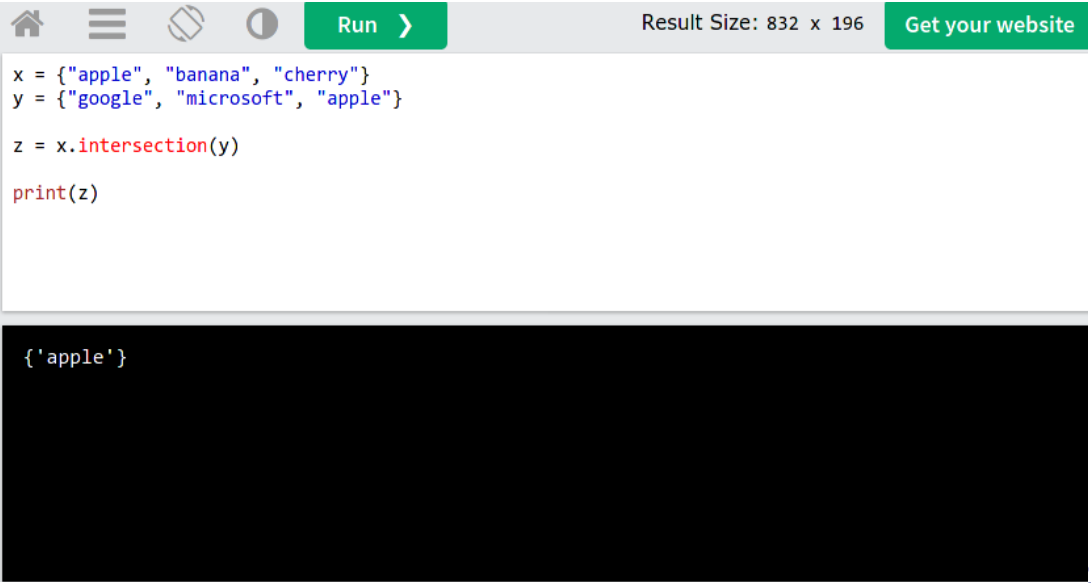

```
S1 = {"apple", "banana", "cherry", 'banana'}
S2 = {"orange","orange"}

S3 = S1.union(S2)

print(S1,S2,S3,sep="\n")

S3.update( {1,2,3,4,5,6} )
print(S3)
```

```
{'banana', 'cherry', 'apple'}
{'orange'}
{'banana', 'cherry', 'orange', 'apple'}
{'banana', 1, 2, 3, 4, 5, 6, 'apple', 'cherry', 'orange'}
```

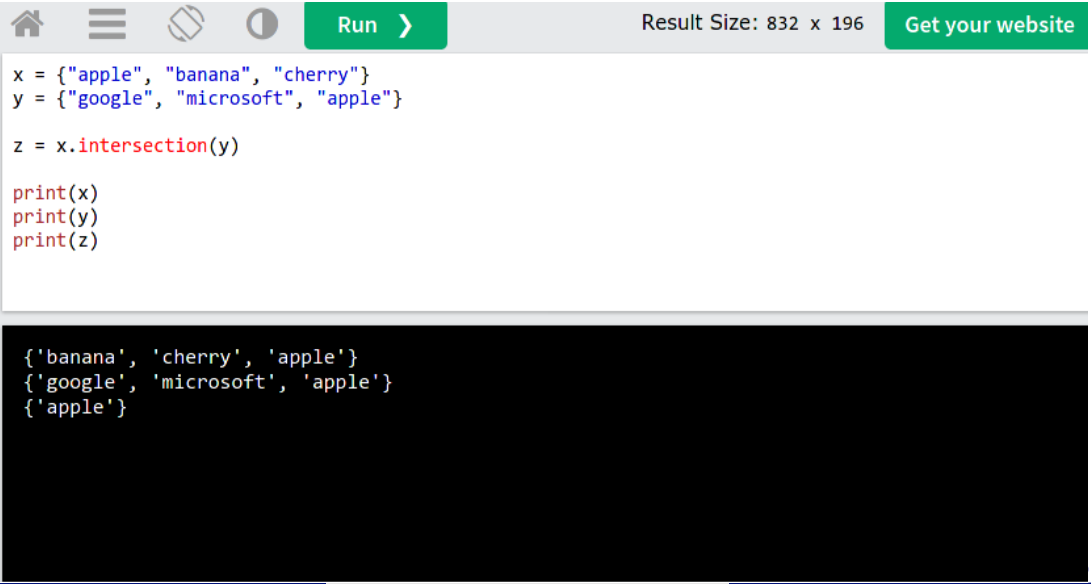
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length

Run ❯ Result Size: 832 x 196 Get your website

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

x.symmetric_difference_update(y)

print(x)
```

```
{'banana', 'microsoft', 'google', 'cherry'}
```

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length




Run ❯ Result Size: 832 x 196 Get your website

```
x = {"apple", "banana", "cherry"}
y = {"google", "microsoft", "apple"}

z = x.symmetric_difference(y)

print(z)
```

```
{'banana', 'cherry', 'microsoft', 'google'}
```

Dicionários

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Dicionários são estruturas de chave-valor, ou seja, os valores (dados) estão sempre associados a uma chave.
- Exemplos de declaração de um dicionário:

```
dicionario = {} # dicionario vazio.
print(type(dicionario))
# <class 'dict'>
```

```
localizacao = {
  "Lat": -22.817087,
  "Long": -47.069750
}
print(type(localizacao))
# <class 'dict'>
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Podemos também declarar um dicionário de maneira explícita utilizando a função `dict`.

```
dicionario = dict({}) # dicionario vazio.
print(type(dicionario))
# <class 'dict'>
```

```
localizacao = dict({
  "Lat": -22.817087,
  "Long": -47.069750
})
print(type(localizacao))
# <class 'dict'>
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Chaves e valores em um dicionário podem ser de diferentes tipos de dados (`int`, `float`, `bool`, entre outros).

```
dicionario = {
  1.2: True,
  123: "Um Dois Três",
  "cat": "dog",
  ("X", "Y"): (2, 3, 5)
}
print(dicionario)
# {1.2: True, 123: 'Um Dois Três', 'cat': 'dog',
#  ('X', 'Y'): (2, 3, 5)}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Geralmente as chaves são mantidas na ordem em que o dicionário é criado ou alterado.

```
dicionario = {
  "Z": 1,
  "A": 2,
  "C": 3
}
print(dicionario)
# {'Z': 1, 'A': 2, 'C': 3}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Podemos acessar um valor do dicionário da seguinte forma:

```
<dicionário>[<chave>]
```

- Essa operação retorna o valor associado à chave informada.
- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print(dicionario["Nome"])
# Ash Ketchum
print(dicionario["Idade"])
# 10
print(dicionario["Profissão"])
# Treinador Pokémon
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- E se informarmos uma chave que não está no dicionário? O que acontece?
- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print(dicionario["Cidade"])
# KeyError: 'Cidade'
```

- Um erro relacionado à chave é gerado.

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Similar ao que vimos em listas, podemos verificar se uma chave está presente ou não em um dicionário utilizando o operador de inclusão `in`.
- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print("Cidade" in dicionario)
# False
print("Nome" in dicionario)
# True
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O método get é outra forma para obter valores de um dicionário. Ele recebe como parâmetro a chave associada ao valor desejado.
- Caso a chave não seja encontrada no dicionário, será retornado None como resposta.
- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print(dicionario.get("Nome"))
# Ash Ketchum
print(dicionario.get("Cidade"))
# None
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O tamanho de um dicionário também pode ser verificado através da função `len`.
- Cada conjunto de chave e valor corresponde a um elemento.
- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
print(len(dicionario))
# 3
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Novos valores podem ser adicionados dinamicamente a um dicionário informando o novo par chave-valor.

```
<dicionário>[<nova_chave>] = <novo_valor>
```

- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon"
}
dicionario["Cidade"] = "Pallet"
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 10, 'Profissão':
#  'Treinador Pokémon', 'Cidade': 'Pallet'}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Para atualizar um valor já existente em um dicionário, basta atribuir à chave o valor atualizado.

```
<dicionário>[<chave>] = <novo_valor>
```

- Exemplo:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 10,
  "Profissão": "Treinador Pokémon",
  "Cidade": "Pallet"
}
dicionario["Idade"] = 12
print(dicionario["Idade"])
# 12
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Para remover um valor (e chave associada) a um dicionário podemos utilizar o método pop.
- O método pop recebe como parâmetro a chave que está associada ao valor que deve ser removido.
- Como resposta o método retorna o valor que foi removido do dicionário.
- Caso a chave informada como parâmetro não esteja no dicionário, um erro será gerado.

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Removendo um valor de um dicionário utilizando o método pop:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 12,
  "Profissão": "Treinador Pokémon",
  "Cidade": "Pallet"
}
profissao = dicionario.pop("Profissão")
print(profissao)
# Treinador Pokémon
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade': 'Pallet'}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Outra forma de remover um valor (e chave associada) em um dicionário é utilizando a declaração `del`.
- A declaração `del` pode ser utilizada da seguinte forma:

```
1 del <dicionario>[<chave>]
```

- Caso a chave informada como parâmetro não esteja no dicionário, um erro será gerado.

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Removendo um valor de um dicionário utilizando a declaração `del`:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 12,
  "Profissão": "Treinador Pokémon",
  "Cidade": "Pallet"
}
del dicionario["Profissão"]
print(dicionario)
# {'Nome': 'Ash Ketchum', 'Idade': 12, 'Cidade': 'Pallet'}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Erro ao tentar remover um valor cuja chave não está no dicionário:

```
dicionario = {
  "Nome": "Ash Ketchum",
  "Idade": 12,
  "Profissão": "Treinador Pokémon",
  "Cidade": "Pallet"
}
cpf = dicionario.pop("CPF")
# KeyError: 'CPF'
del dicionario["CNH"]
# KeyError: 'CNH'
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O método popitem remove sempre o último par (chave, valor) de um dicionário.
- Como resposta o método retorna o par (chave, valor) removido, em formato de tupla.
- Caso o dicionário esteja vazio, um erro será gerado.
- Exemplo:

```
carro = {"Modelo": "Gol", "Ano": 2019}
print(carro)
# {'Modelo': 'Gol', 'Ano': 2019}
print(carro.popitem())
# ('Ano', 2019)
print(carro.popitem())
# ('Modelo', 'Gol')
print(carro)
# {}
print(carro.popitem())
# KeyError: 'popitem(): dictionary is empty'
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O método update pode ser utilizado para atualizar um dicionário.
- Esse método recebe como parâmetro um outro dicionário.
- O dicionário original é modificado com base no dicionário informado como parâmetro, de tal forma que os valores das chaves previamente existentes no primeiro são atualizados e novos valores são adicionados para as novas chaves.

```
1 dic_a = {"A": "Avião", "B": "Barco"}
2 dic_b = {"B": "Balão", "C": "Carro"}
3 dic_a.update(dic_b)
4 print(dic_a)
5 # {'A': 'Avião', 'B': 'Balão', 'C': 'Carro'}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O método `keys` retorna uma estrutura com as chaves do dicionário, que pode ser convertida para uma lista.

```
lugar = {"Lat": -22.817087, "Long": -47.069750}
print(lugar.keys())
# dict_keys(['Lat', 'Long'])
print(list(lugar.keys()))
# ['Lat', 'Long']
```

- O método `values` retorna uma estrutura com os valores do dicionário, que pode ser convertida para uma lista.

```
lugar = {"Lat": -22.817087, "Long": -47.069750}
print(lugar.values())
# dict_values([-22.817087, -47.06975])
print(list(lugar.values()))
# [-22.817087, -47.06975]
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- O método `items` retorna uma estrutura que pode ser convertida para uma lista de tuplas, onde cada tupla é composta pelo par (chave, valor).

```
1 lugar = {"Lat": -22.817087, "Long": -47.069750}
2 print(lugar.items())
3 # dict_items([('Lat', -22.817087), ('Long', -47.06975)])
4 print(list(lugar.items()))
5 # [('Lat', -22.817087), ('Long', -47.06975)]
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Podemos iterar sobre uma lista de chaves utilizando o método `keys`.
- Exemplo:

```
dic = {
  "A": "Abacate",
  "B": "Banana",
  "C": "Caqui"
}
for letra in dic.keys():
  print("Letra:", letra)
# Letra: A
# Letra: B
# Letra: C
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Podemos iterar sobre uma lista de valores utilizando o método `values`.
- Exemplo:

```
dic = {
  "A": "Abacate",
  "B": "Banana",
  "C": "Caqui"
}
for fruta in dic.values():
  print("Fruta:", fruta)
# Fruta: Abacate
# Fruta: Banana
# Fruta: Caqui
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Podemos também iterar sobre uma lista de tuplas contendo as chaves e os valores utilizando o método `items`.
- Exemplo:

```
dic = {
  "A": "Abacate",
  "B": "Banana",
  "C": "Caqui"
}
for (letra, fruta) in dic.items():
  print("Fruta com Letra", letra, ":", fruta)
# Fruta com Letra A: Abacate
# Fruta com Letra B: Banana
# Fruta com Letra C: Caqui
```

Descrição

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

Escreva um programa que dada a lista a seguir:

```
dados = [
  {"dia": 12, "mes": 2, "ano": 2019, "temp": 30.5},
  {"dia": 18, "mes": 3, "ano": 2019, "temp": 29.1},
  {"dia": 22, "mes": 4, "ano": 2019, "temp": 28.5},
  {"dia": 17, "mes": 5, "ano": 2019, "temp": 26.4}
]
```

... imprime como resposta a seguinte saída:

```
# 12/02/2019: Temperatura: 30.5C
# 18/03/2019: Temperatura: 29.1C
# 22/04/2019: Temperatura: 28.5C
# 17/05/2019: Temperatura: 26.4C
```

Seu código deve iterar sobre a lista acessando cada dicionário.

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

Resposta

Uma possível resposta para o exercício:

```
dados = [
  {"dia": 12, "mes": 2, "ano": 2019, "temp": 30.5},
  {"dia": 18, "mes": 3, "ano": 2019, "temp": 29.1},
  {"dia": 22, "mes": 4, "ano": 2019, "temp": 28.5},
  {"dia": 17, "mes": 5, "ano": 2019, "temp": 26.4}
]

msg = "{0:02d}/{1:02d}/{2}: Temperatura: {3}C"
for dic in dados:
  print(msg.format(dic["dia"], dic["mes"],
                   dic["ano"], dic["temp"]))
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Similar ao que vimos em listas, podemos atribuir um dicionário para diferentes variáveis, mas as variáveis estarão relacionadas ao mesmo dicionário (objeto).
- Exemplo:

```
dic_a = {"Nome": "João", "Idade": 18}
print(dic_a)
# {'Nome': 'João', 'Idade': 18}
dic_b = dic_a
dic_b["Nome"] = "Maria"
print(dic_b)
# {'Nome': 'Maria', 'Idade': 18}
print(dic_a)
# {'Nome': 'Maria', 'Idade': 18}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- Similar ao que vimos em listas, se quisermos criar uma cópia independente de um dicionário devemos utilizar o método copy.
- Exemplo:

```
dic_a = {"Nome": "João", "Idade": 18}
print(dic_a)
# {'Nome': 'João', 'Idade': 18}
dic_b = dic_a.copy()
dic_b["Nome"] = "Maria"
print(dic_b)
# {'Nome': 'Maria', 'Idade': 18}
print(dic_a)
# {'Nome': 'João', 'Idade': 18}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- É possível criar um dicionário a partir de duas listas com o auxílio da função `zip`.
- A função `zip` recebe dois parâmetros, o primeiro é uma lista contendo as chaves desejadas para o dicionário, enquanto o segundo é uma lista contendo os respectivos valores.
- Exemplo:

```
pessoas = ["Alice", "Beatriz", "Carlos"]
telefones = ["99999-0000", "99999-1111", "99999-2222"]
contatos = dict(zip(pessoas, telefones))
print(contatos)
# {'Alice': '99999-0000',
#  'Beatriz': '99999-1111',
#  'Carlos': '99999-2222'}
```

https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- É possível criar um dicionário cujos valores são listas.
- Exemplo:

```
pessoas = {}
n = int(input())
for i in range(n):
  nome = input()
  idade = int(input())
  sexo = input()
  pessoas[nome] = [idade, sexo]
print(pessoas)
# {'Alice': [20, 'F'],
#  'Beatriz': [18, 'F'],
#  'Carlos ': [19, 'M']}
print(pessoas["Beatriz"][1])
# F
```

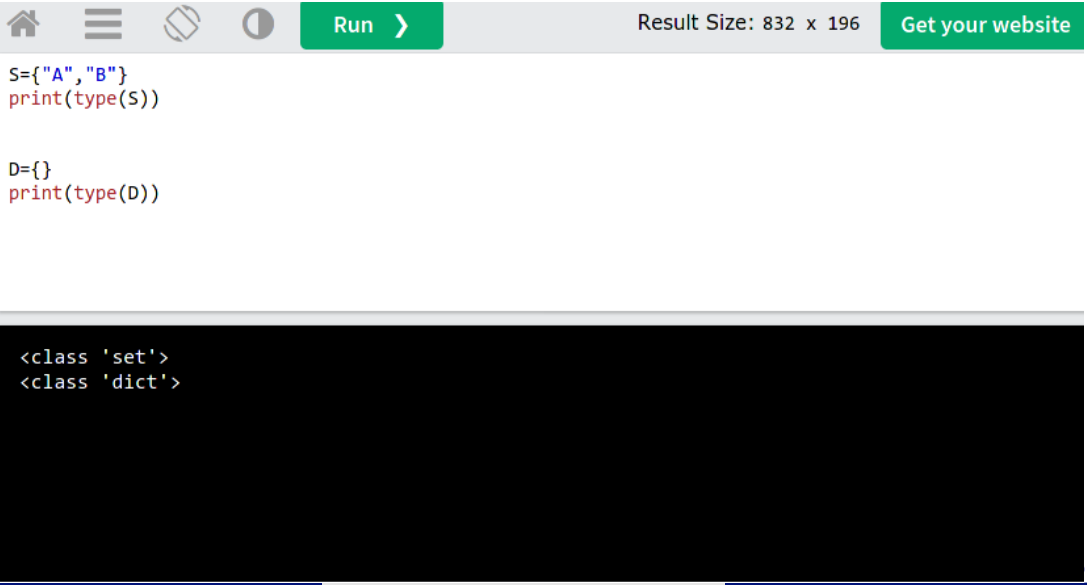
https://ic.unicamp.br/~mc102/aulas/aula07.pdf

- É possível criar um dicionário cujos valores são dicionários.
- Exemplo:

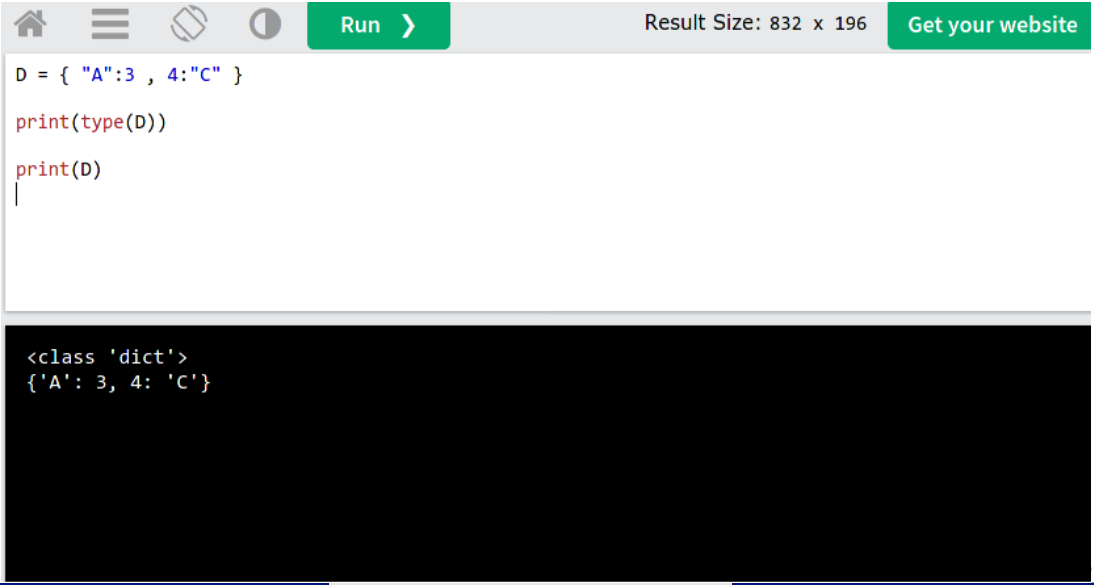
```
pessoas = {}
n = int(input())
for i in range(n):
  nome = input()
  idade = int(input())
  sexo = input()
  pessoas[nome] = {"idade": idade, "sexo": sexo}
print(pessoas)
# {'Alice': {'idade': 20, 'sexo': 'F'},
#  'Beatriz': {'idade': 18, 'sexo': 'F'},
#  'Carlos ': {'idade': 19, 'sexo': 'M'}}
print(pessoas["Beatriz"]["sexo"])
# F
```

Exemplos usando Dicionários

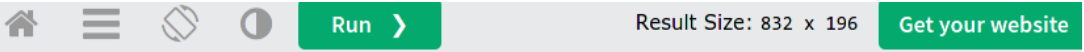
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



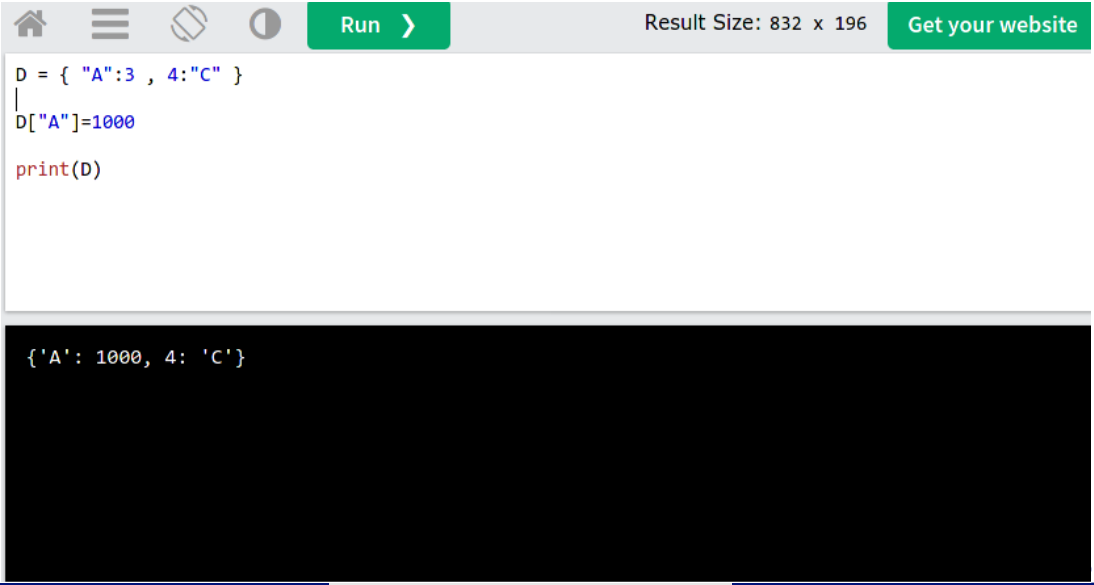
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



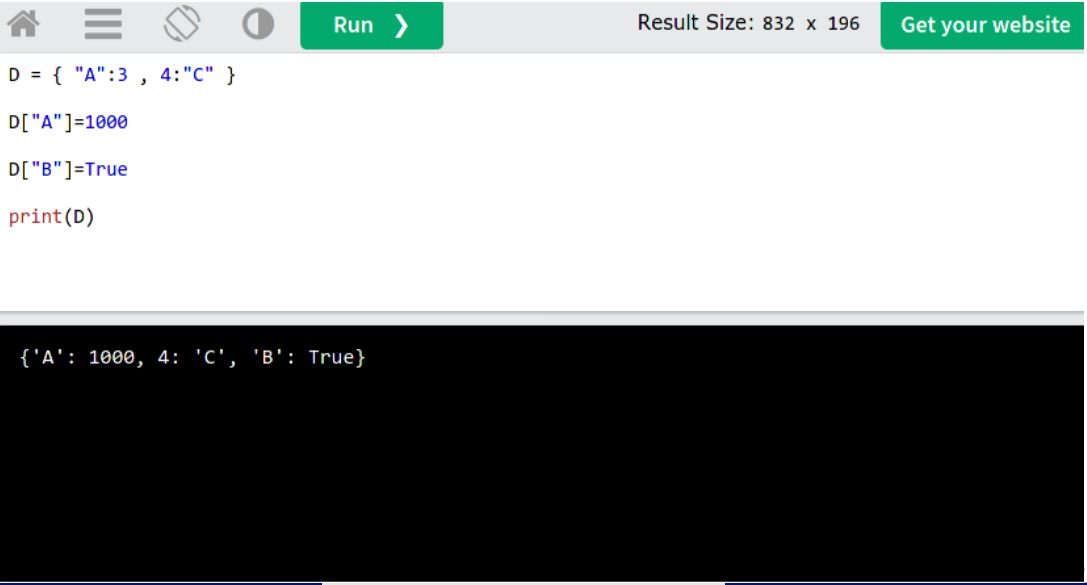

```
D = { "A":3 , 4:"C" }

print(type(D))

print(D)

print(D["A"])
print(D[4])
```

```
<class 'dict'>
{'A': 3, 4: 'C'}
3
C
```

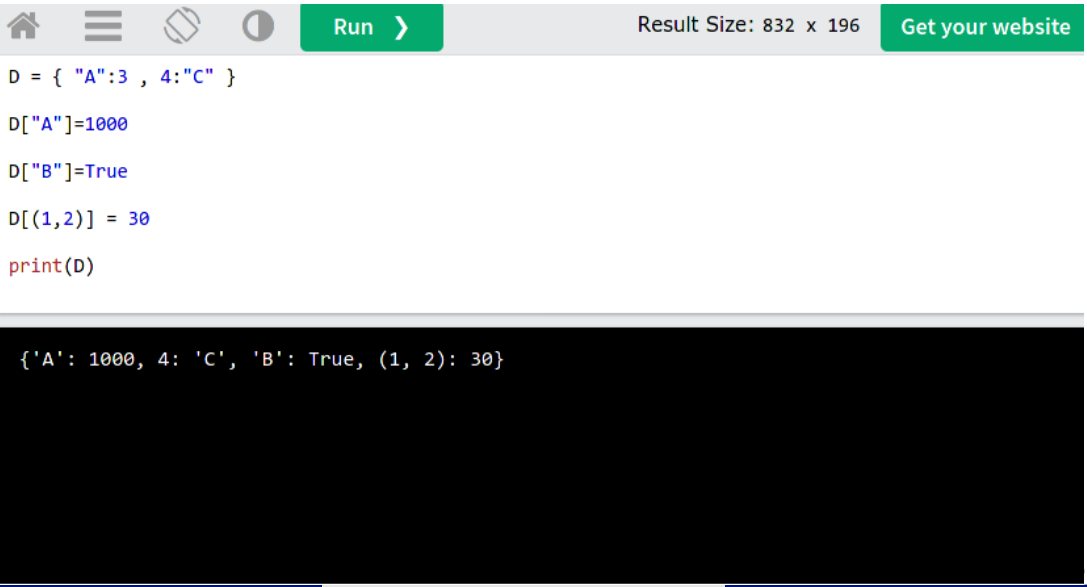
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



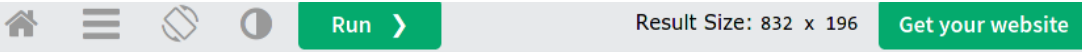
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



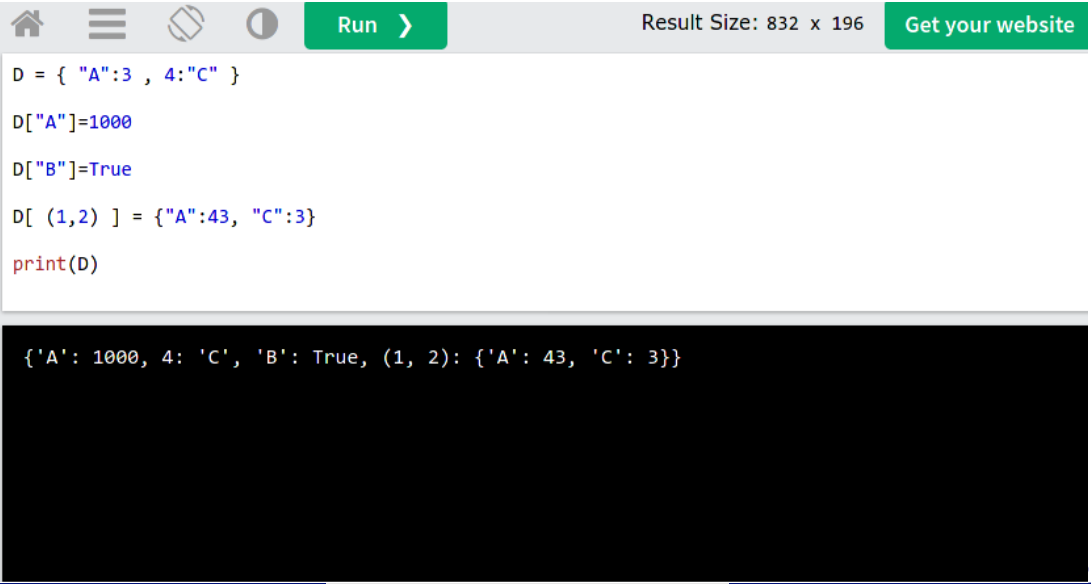
https://www.w3schools.com/python/trypython.asp?filename=demo_set_length




```
D = { "A":3 , 4:"C" }

D["A"]=1000

D["B"]=True

D[ [1,2] ] = 30

print(D)
```

```
Traceback (most recent call last):
  File "./prog.py", line 7, in <module>
TypeError: unhashable type: 'list'
```

https://www.w3schools.com/python/trypython.asp?filename=demo_set_length



Perguntas ....

Referências

- Zanoni Dias, MC102, Algoritmos e Programação de Computadores, IC/UNICAMP, 2021. https://ic.unicamp.br/~mc102/

  Aula Introdutória [slides] [vídeo]
  Primeira Aula de Laboratório [slides] [vídeo]
  Python Básico: Tipos, Variáveis, Operadores, Entrada e Saída [slides] [vídeo]
  Comandos Condicionais [slides] [vídeo]
  Comandos de Repetição [slides] [vídeo]
  Listas e Tuplas [slides] [vídeo]
  Strings [slides] [vídeo]
  Dicionários [slides] [vídeo]
  Funções [slides] [vídeo]
  Objetos Multidimensionais [slides] [vídeo]
  Algoritmos de Ordenação [slides] [vídeo]
  Algoritmos de Busca [slides] [vídeo]
  Recursão [slides] [vídeo]
  Algoritmos de Ordenação Recursivos [slides] [vídeo]
  Arquivos [slides] [vídeo]
  Expressões Regulares [slides] [vídeo]
  Execução de Testes no Google Cloud Shell [slides] [vídeo]
  Numpy [slides] [vídeo]
  Pandas [slides] [vídeo]

- Panda - Cursos de Computação em Python (IME -USP) https://panda.ime.usp.br/

  Como Pensar Como um Cientista da Computação https://panda.ime.usp.br/pensepy/static/pensepy/
  Aulas de Introdução à Computação em Python https://panda.ime.usp.br/aulasPython/static/aulasPython/

- Fabio Kon, Introdução à Ciência da Computação com Python http://bit.ly/FabioKon/
- Socratica, Python Programming Tutorials http://bit.ly/SocraticaPython/
- Google - online editor for cloud-native applications (Python programming) https://shell.cloud.google.com/
- w3schools - Python Tutorial https://www.w3schools.com/python/
- Outros, citados nos Slides.