Python

2023-12-29

Contents

1 VARIABLES AND SIMPLE DATA TYPES

- Changing Case in a String with Methods

```
# Escribe la primera letra de cada palabra en mayúscula
name = "ada lovelace"
print(name.title())
```

```
## Ada Lovelace
```

```
# Escribe toda la palabra en mayúsculas
print(name.upper())
```

```
## ADA LOVELACE
```

```
# Escribe toda la palabra en minúsculas
print(name.lower())
```

```
## ada lovelace
```

- Using Variables in Strings

```
first_name = "ada"
last_name = "lovelace"
full_name = f"{first_name} {last_name}"
print(full_name)
```

```
## ada lovelace
```

- Adding Whitespace to Strings with Tabs or Newlines

```
print("Languages:\n\tPython\n\tC\n\tJavaScript")
```

```
## Languages:
##  Python
##  C
##  JavaScript
```

- Stripping Whitespace

```
favorite_language = ' python '
favorite_language.rstrip()
```

```
## ' python'
```

```
favorite_language.lstrip()
```

```
## 'python '
```

```
favorite_language.strip()
```

```
## 'python'
```

- Removing Prefixes

```
nostarch_url = 'https://nostarch.com'
nostarch_url.removeprefix('https://')
```

```
## 'nostarch.com'
```

- Underscores in Numbers

```
universe_age = 14_000_000_000
print(universe_age)
```

```
## 14000000000
```

- Multiple Assignment

```
x, y, z = 0, 0, 0
```

2 INTRODUCING LISTS

```
bicycles = ['trek', 'cannondale', 'redline', 'specialized']
print(bicycles)
```

```
## ['trek', 'cannondale', 'redline', 'specialized']
```

- Accessing Elements in a List

```
print(bicycles[0].title())
```

```
## Trek
```

Python has a special syntax for accessing the last element in a list. If you ask for the item at index -1, Python always returns the last item in the list:

```
print(bicycles[-1])
```

```
## specialized
```

- Using Individual Values from a List

```
message = f"My first bicycle was a {bicycles[0].title()}."
print(message)
```

```
## My first bicycle was a Trek.
```

- Modifying Elements in a List

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki']
```

```
motorcycles[0] = 'ducati'
print(motorcycles)
```

```
## ['ducati', 'yamaha', 'suzuki']
```

- Adding Elements to a List

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki']
```

```
motorcycles.append('ducati')
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki', 'ducati']
```

- Inserting Elements into a List

```
motorcycles = ['honda', 'yamaha', 'suzuki']
motorcycles.insert(0, 'ducati')
print(motorcycles)
```

```
## ['ducati', 'honda', 'yamaha', 'suzuki']
```

- Removing an Item Using the del Statement

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki']
```

```
del motorcycles[0]
print(motorcycles)
```

```
## ['yamaha', 'suzuki']
```

- Removing an Item Using the pop() Method

```
motorcycles = ['honda', 'yamaha', 'suzuki']
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki']
```

```
popped_motorcycle = motorcycles.pop()
print(motorcycles)
```

```
## ['honda', 'yamaha']
```

```
print(popped_motorcycle)
```

```
## suzuki
```

- Popping Items from Any Position in a List

```
motorcycles = ['honda', 'yamaha', 'suzuki']
first_owned = motorcycles.pop(0)
print(f"The first motorcycle I owned was a {first_owned.title()}.")
```

```
## The first motorcycle I owned was a Honda.
```

- Removing an Item by Value

```
motorcycles = ['honda', 'yamaha', 'suzuki', 'ducati']
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki', 'ducati']
```

```
motorcycles.remove('ducati')
print(motorcycles)
```

```
## ['honda', 'yamaha', 'suzuki']
```

- Sorting a List Permanently with the sort() Method

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
cars.sort()
print(cars)
```

```
## ['audi', 'bmw', 'subaru', 'toyota']
```

- Sorting a List Temporarily with the sorted() Function

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print("Here is the original list:")
```

```
## Here is the original list:
```

```
print(cars)
```

```
## ['bmw', 'audi', 'toyota', 'subaru']
```

```
print("\nHere is the sorted list:")
```

```
##
## Here is the sorted list:
```

```
print(sorted(cars))
```

```
## ['audi', 'bmw', 'subaru', 'toyota']
```

```
print("\nHere is the original list again:")
```

```
##
## Here is the original list again:
```

```
print(cars)
```

```
## ['bmw', 'audi', 'toyota', 'subaru']
```

- Printing a List in Reverse Order

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
print(cars)
```

```
## ['bmw', 'audi', 'toyota', 'subaru']
```

```
cars.reverse()
print(cars)
```

```
## ['subaru', 'toyota', 'audi', 'bmw']
```

- Finding the Length of a List

```
cars = ['bmw', 'audi', 'toyota', 'subaru']
len(cars)
```

```
## 4
```

3 WORKING WITH LISTS

- Looping Through an Entire List

```
magicians = ['alice', 'david', 'carolina']
for magician in magicians:
  print(magician)
```

```
## alice
## david
## carolina
```

- Using the range() Function

```
for value in range(1, 5):
  print(value)
```

```
## 1
## 2
## 3
```

```
## 4
```

- Using range() to Make a List of Numbers

```
numbers = list(range(1, 6))
print(numbers)
```

```
## [1, 2, 3, 4, 5]
```

```
even_numbers = list(range(2, 11, 2))
print(even_numbers)
```

```
## [2, 4, 6, 8, 10]
```

```
squares = []
for value in range(1,11):
  squares.append(value**2)
print(squares)
```

```
## [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- Simple Statistics with a List of Numbers

```
digits = [1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
min(digits)
```

```
## 0
```

```
max(digits)
```

```
## 9
```

```
sum(digits)
```

```
## 45
```

- List Comprehensions

```
squares = [value**2 for value in range(1, 11)]
print(squares)
```

```
## [1, 4, 9, 16, 25, 36, 49, 64, 81, 100]
```

- Slicing a List

```
players = ['charles', 'martina', 'michael', 'florence', 'eli']
print(players[0:3])
```

```
## ['charles', 'martina', 'michael']
```

```
print(players[:4])
```

```
## ['charles', 'martina', 'michael', 'florence']
```

```
print(players[2:])
```

```
## ['michael', 'florence', 'eli']
```

```
# con el signo negativo cuenta desde el final
print(players[-3:])
```

```
## ['michael', 'florence', 'eli']
```

- Looping Through a Slice

```
print("Here are the first three players on my team:")
```

```
## Here are the first three players on my team:
```

```
for player in players[:3]:
  print(player.title())
```

```
## Charles
## Martina
## Michael
```

- Copying a List

```
my_foods = ['pizza', 'falafel', 'carrot cake']
friend_foods = my_foods[:]
print("My favorite foods are:")
```

```
## My favorite foods are:
```

```
print(my_foods)
```

```
## ['pizza', 'falafel', 'carrot cake']
```

```
print("\nMy friend's favorite foods are:")
```

```
##
## My friend's favorite foods are:
```

```
print(friend_foods)
```

```
## ['pizza', 'falafel', 'carrot cake']
```

- Defining a Tuple Las tuplas son como las listas, pero no se pueden modificar. Si queremos cambiar una tupla tenemos que redefinirla.

```
dimensions = (200, 50)
print(dimensions[0])
```

```
## 200
```

```
print(dimensions[1])
```

```
## 50
```

4 IF STATEMENTS

- A Simple Example

```
cars = ['audi', 'bmw', 'subaru', 'toyota']
for car in cars:
  if car == 'bmw':
    print(car.upper())
  else:
    print(car.title())
```

```
## Audi
## BMW
## Subaru
## Toyota
```

- Checking for Inequality

```
requested_topping = 'mushrooms'
if requested_topping != 'anchovies':
  print("Hold the anchovies!")
```

```
## Hold the anchovies!
```

- Numerical Comparisons

```
answer = 17
if answer != 42:
  print("That is not the correct answer. Please try again!")
```

```
## That is not the correct answer. Please try again!
```

- Checking Whether a Value Is in a List

```
requested_toppings = ['mushrooms', 'onions', 'pineapple']
'mushrooms' in requested_toppings
```

```
## True
```

```
'pepperoni' in requested_toppings
```

```
## False
```

- Checking Whether a Value Is Not in a List

```
banned_users = ['andrew', 'carolina', 'david']
user = 'marie'
if user not in banned_users:
  print(f"{user.title()}, you can post a response if you wish.")
```

```
## Marie, you can post a response if you wish.
```

- Testing Multiple Conditions The if- elif- else block would stop running after only one test passes.
- Using if Statements with Lists

```
requested_toppings = ['mushrooms', 'green peppers', 'extra cheese']
for requested_topping in requested_toppings:
  print(f"Adding {requested_topping}.")
```

```
## Adding mushrooms.
## Adding green peppers.
## Adding extra cheese.
```

```
print("\nFinished making your pizza!")
```

```
##
## Finished making your pizza!
```

```
for requested_topping in requested_toppings:
  if requested_topping == 'green peppers':
    print("Sorry, we are out of green peppers right now.")
  else:
    print(f"Adding {requested_topping}.")
```

```
## Adding mushrooms.
## Sorry, we are out of green peppers right now.
## Adding extra cheese.
```

```
print("\nFinished making your pizza!")
```

```
##
## Finished making your pizza!
```

- Checking That a List Is Not Empty

```
requested_toppings = []
if requested_toppings:
  for requested_topping in requested_toppings:
    print(f"Adding {requested_topping}.")
  print("\nFinished making your pizza!")
else:
  print("Are you sure you want a plain pizza?")
```

```
## Are you sure you want a plain pizza?
```

- Using Multiple Lists

```
available_toppings = ['mushrooms', 'olives', 'green peppers',
'pepperoni', 'pineapple', 'extra cheese']
requested_toppings = ['mushrooms', 'french fries', 'extra cheese']
for requested_topping in requested_toppings:
  if requested_topping in available_toppings:
    print(f"Adding {requested_topping}.")
  else:
    print(f"Sorry, we don't have {requested_topping}.")
```

```
## Adding mushrooms.
## Sorry, we don't have french fries.
## Adding extra cheese.
```

```
print("\nFinished making your pizza!")
```

```
##
## Finished making your pizza!
```

5 DICTIONARIES

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0['color'])
```

```
## green
```

```
print(alien_0['points'])
```

```
## 5
```

- Adding New Key-Value Pairs

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0)
```

```
## {'color': 'green', 'points': 5}
```

```
alien_0['x_position'] = 0
alien_0['y_position'] = 25
print(alien_0)
```

```
## {'color': 'green', 'points': 5, 'x_position': 0, 'y_position': 25}
```

- Starting with an Empty Dictionary

```
alien_0 = {}
alien_0['color'] = 'green'
alien_0['points'] = 5
print(alien_0)
```

```
## {'color': 'green', 'points': 5}
```

- Modifying Values in a Dictionary

```
alien_0 = {'color': 'green'}
print(f"The alien is {alien_0['color']}.")
```

```
## The alien is green.
```

```
alien_0['color'] = 'yellow'
print(f"The alien is now {alien_0['color']}.")
```

```
## The alien is now yellow.
```

- Removing Key-Value Pairs

```
alien_0 = {'color': 'green', 'points': 5}
print(alien_0)
```

```
## {'color': 'green', 'points': 5}
```

```
del alien_0['points']
print(alien_0)
```

```
## {'color': 'green'}
```

- Using get() to Access Values

  Using keys in square brackets to retrieve the value you're interested in from a dictionary might cause one potential problem: if the key you ask for doesn't exist, you'll get an error.

  The get() method requires a key as a first argument. As a second optional argument, you can pass the value to be returned if the key doesn't exist:

```
alien_0 = {'color': 'green', 'speed': 'slow'}
point_value = alien_0.get('points', 'No point value assigned.')
print(point_value)
```

```
## No point value assigned.
```

- Looping Through a Dictionary

```
user_0 = {
  'username': 'efermi',
  'first': 'enrico',
  'last': 'fermi',
}

for key, value in user_0.items():
  print(f"\nKey: {key}")
  print(f"Value: {value}")
```

```
##
## Key: username
```

```
## Value: efermi
##
## Key: first
## Value: enrico
##
## Key: last
## Value: fermi
```

- Looping Through All the Keys in a Dictionary

```
favorite_languages = {
  'jen': 'python',
  'sarah': 'c',
  'edward': 'rust',
  'phil': 'python',
}

for name in favorite_languages.keys():
  print(name.title())
```

```
## Jen
## Sarah
## Edward
## Phil
```

Looping through the keys is actually the default behavior when looping through a dictionary, so this code would have exactly the same output if you wrote:

```
for name in favorite_languages:
  print(name.title())
```

```
## Jen
## Sarah
## Edward
## Phil
```

You can access the value associated with any key you care about inside the loop, by using the current key. Let's print a message to a couple of friends about the languages they chose. We'll loop through the names in the dictionary as we did previously, but when the name matches one of our friends, we'll display a message about their favorite language:

```
friends = ['phil', 'sarah']
for name in favorite_languages.keys():
  print(f"Hi {name.title()}.")

  if name in friends:
    language = favorite_languages[name].title()
    print(f"\t{name.title()}, I see you love {language}!")
```

```
## Hi Jen.
## Hi Sarah.
##  Sarah, I see you love C!
## Hi Edward.
## Hi Phil.
##  Phil, I see you love Python!
```

You can also use the keys() method to find out if a particular person was polled. This time, let's find out if Erin took the poll:

```
if 'erin' not in favorite_languages.keys():
  print("Erin, please take our poll!")
```

```
## Erin, please take our poll!
```

- Looping Through a Dictionary's Keys in a Particular Order

```
for name in sorted(favorite_languages.keys()):
  print(f"{name.title()}, thank you for taking the poll.")
```

```
## Edward, thank you for taking the poll.
## Jen, thank you for taking the poll.
## Phil, thank you for taking the poll.
## Sarah, thank you for taking the poll.
```

- Looping Through All Values in a Dictionary

```
print("The following languages have been mentioned:")
```

```
## The following languages have been mentioned:
```

```
for language in favorite_languages.values():
  print(language.title())
```

```
## Python
## C
## Rust
## Python
```

This approach pulls all the values from the dictionary without checking for repeats. This might work fine with a small number of values, but in a poll with a large number of respondents, it would result in a very repetitive list. To see each language chosen without repetition, we can use a set. A set is a collection in which each item must be unique:

```
print("The following languages have been mentioned:")
```

```
## The following languages have been mentioned:
```

```
for language in set(favorite_languages.values()):
  print(language.title())
```

```
## Rust
## C
## Python
```

You can build a set directly using braces and separating the elements with commas:

```
languages = {'python', 'rust', 'python', 'c'}
languages
```

```
## {'rust', 'c', 'python'}
```

- A List of Dictionaries

```
alien_0 = {'color': 'green', 'points': 5}
alien_1 = {'color': 'yellow', 'points': 10}
alien_2 = {'color': 'red', 'points': 15}
aliens = [alien_0, alien_1, alien_2]
for alien in aliens:
  print(alien)
```

```
## {'color': 'green', 'points': 5}
## {'color': 'yellow', 'points': 10}
## {'color': 'red', 'points': 15}
```

A more realistic example would involve more than three aliens with code that automatically generates each alien. In the following example, we use range() to create a fleet of 30 aliens:

```
# Make an empty list for storing aliens.
aliens = []

# Make 30 green aliens.
for alien_number in range(30):
  new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
  aliens.append(new_alien)

# Show the first 5 aliens.
for alien in aliens[:5]:
  print(alien)
```

```
## {'color': 'green', 'points': 5, 'speed': 'slow'}
## {'color': 'green', 'points': 5, 'speed': 'slow'}
## {'color': 'green', 'points': 5, 'speed': 'slow'}
## {'color': 'green', 'points': 5, 'speed': 'slow'}
## {'color': 'green', 'points': 5, 'speed': 'slow'}
```

```
print("...")
```

```
## ...
```

```
# Show how many aliens have been created.
print(f"Total number of aliens: {len(aliens)}")
```

```
## Total number of aliens: 30
```

We can use a for loop and an if statement to change the color of the aliens. For example, to change the first three aliens to yellow, medium-speed aliens worth 10 points each, we could do this:

```
# Make an empty list for storing aliens.
aliens = []

# Make 30 green aliens.
for alien_number in range (30):
  new_alien = {'color': 'green', 'points': 5, 'speed': 'slow'}
  aliens.append(new_alien)

for alien in aliens[:3]:
  if alien['color'] == 'green':
    alien['color'] = 'yellow'
    alien['speed'] = 'medium'
    alien['points'] = 10

# Show the first 5 aliens.
for alien in aliens[:5]:
  print(alien)
```

```
## {'color': 'yellow', 'points': 10, 'speed': 'medium'}
## {'color': 'yellow', 'points': 10, 'speed': 'medium'}
## {'color': 'yellow', 'points': 10, 'speed': 'medium'}
```

```
## {'color': 'green', 'points': 5, 'speed': 'slow'}
## {'color': 'green', 'points': 5, 'speed': 'slow'}
```

```
print("...")
```

```
## ...
```

- A List in a Dictionary

```
# Store information about a pizza being ordered.
pizza = {
  'crust': 'thick',
  'toppings': ['mushrooms', 'extra cheese'],
}

# Summarize the order.
print(f"You ordered a {pizza['crust']}-crust pizza "
  "with the following toppings:")
```

```
## You ordered a thick-crust pizza with the following toppings:
```

```
for topping in pizza['toppings']:
  print(f"\t{topping}")
```

```
##  mushrooms
##  extra cheese
```

You can nest a list inside a dictionary anytime you want more than one value to be associated with a single key in a dictionary.

```
favorite_languages = {
  'jen': ['python', 'rust'],
  'sarah': ['c'],
  'edward': ['rust', 'go'],
  'phil': ['python', 'haskell'],
}

for name, languages in favorite_languages.items():
  print(f"\n{name.title()}'s favorite languages are:")
  for language in languages:
    print(f"\t{language.title()}")
```

```
##
## Jen's favorite languages are:
##  Python
##  Rust
##
## Sarah's favorite languages are:
##  C
##
## Edward's favorite languages are:
##  Rust
##  Go
##
## Phil's favorite languages are:
##  Python
##  Haskell
```

- A Dictionary in a Dictionary

```
users = {
  'aeinstein': {
    'first': 'albert',
    'last': 'einstein',
    'location': 'princeton',
  },

  'mcurie': {
    'first': 'marie',
    'last': 'curie',
    'location': 'paris',
  },
}

for username, user_info in users.items():
  print(f"\nUsername: {username}")
  full_name = f"{user_info['first']} {user_info['last']}"
  location = user_info['location']

  print(f"\tFull name: {full_name.title()}")
  print(f"\tLocation: {location.title()}")
```

```
##
## Username: aeinstein
##  Full name: Albert Einstein
##  Location: Princeton
##
## Username: mcurie
##  Full name: Marie Curie
##  Location: Paris
```

6 USER INPUT AND WHILE LOOPS

- How the input() Function Works

```
# message = input("Tell me something, and I will repeat it back to you: ")
# print(message)
```

- Writing Clear Prompts

```
# name = input("Please enter your name: ")
#print(f"\nHello, {name}!")
```

Sometimes you'll want to write a prompt that's longer than one line.

```
# prompt = "If you share your name, we can personalize the messages you see."
# prompt += "\nWhat is your first name? "

# name = input(prompt)
# print(f"\nHello, {name}!")
```

- Using int() to Accept Numerical Input

```
# height = input("How tall are you, in inches? ")
# height = int(height)
```

```
# if height >= 48:
  # print("\nYou're tall enough to ride!")
# else:
  # print("\nYou'll be able to ride when you're a little older.")
```

- The Modulo Operator

```
# number = input("Enter a number, and I'll tell you if it's even or odd: ")
# number = int(number)

# if number % 2 == 0:
  # print(f"\nThe number {number} is even.")
# else:
  # print(f"\nThe number {number} is odd.")
```

- The while Loop in Action