**The for Loop**

Najczęstszym zastosowaniem pętli jest to, kiedy chcemy wykonać jakiś kod dla każdej pozycji w sekwencji. W przypadku tego typu pętli lub iteracji użyjemy forloopa. Ogólna struktura pętli for jest:

for TEMP\_VAR in SEQUENCE:

pass

TEMP\_VAR zostanie zapełniony każdym elementem, gdy iterujemy przez SEQUENCE i będzie on dostępny dla nas w kontekście pętli. Gdy pętla zakończy jedną iterację, TEMP\_VAR zostanie zapełniony następnym elementem w SEQUENCE, a treść pętli zostanie wykonana ponownie. Proces ten jest kontynuowany, dopóki nie trafimy na instrukcję przerwania lub nie będziemy powtarzać każdego elementu w SEKWENCJI. Oto przykład zapętlania listy kolorów:

>>> colors = ['blue', 'green', 'red', 'purple']

>>> for color in colors:

... print(color)

...

blue

green

red

purple

>>> color

'purple'

Gdybyśmy nie chcieli drukować pewnych kolorów, moglibyśmy ponownie użyć instrukcji kontynuuj lub łamaj. Powiedzmy, że chcemy pominąć ciąg „niebieski” i zakończyć pętlę, jeśli zobaczymy ciąg „czerwony”:

>>> colors = ['blue', 'green', 'red', 'purple']

>>> for color in colors:

... if color == 'blue':

... continue

... elif color == 'red':

... break

... print(color)

...

green

>>>

**Inne typy Iterable**

Listy będą najczęstszym typem, który powtarzamy za pomocą pętli for, ale możemy również iterować po innych typach sekwencji. Z typów, które już znamy, możemy iterować po łańcuchach, słownikach i krotkach.

Oto przykład krotki:

>>> point = (2.1, 3.2, 7.6)

>>> for value in point:

... print(value)

...

2.1

3.2

7.6

>>>

Przykład słownika:

>>> ages = {'kevin': 59, 'bob': 40, 'kayla': 21}

>>> for key in ages:

... print(key)

...

kevin

bob

kayla

Przykład ciągu:

>>> for letter in "my\_string":

... print(letter)

...

m

y

\_

s

t

r

i

n

g

>>>

**Unpacking Multiple Items in a for Loop**

W wideo krotek omówiliśmy, jak można rozdzielić krotkę na wiele zmiennych, „rozpakowując” wartości. Rozpakowywanie działa w kontekście definicji pętli i musisz to wiedzieć, aby najskuteczniej iterować słowniki, ponieważ zazwyczaj potrzebujesz klucza i wartości. Powtórzmy listę „punktów”, aby to sprawdzić:

>>> list\_of\_points = [(1, 2), (2, 3), (3, 4)]

>>> for x, y in list\_of\_points:

... print(f"x: {x}, y: {y}")

...

x: 1, y: 2

x: 2, y: 3

x: 3, y: 4

Widząc, jak działa to rozpakowywanie, skorzystajmy z metody elementów w naszym słowniku wieków, aby wypisać nazwy i wiek:

>>> for name, age in ages.items():

... print(f"Person Named: {name}")

... print(f"Age of: {age}")

...

Person Named: kevin

Age of: 59

Person Named: bob

Age of: 40

Person Named: kayla

Age of: 21