Operatory zasięgu i pętle

W poprzedniej części przyjrzeliśmy się warunkom, które pozwalają robić różne rzeczy w oparciu o różne warunki, oraz opcjonalnym, które umożliwiają tworzenie zmiennych, które mogą, ale nie muszą mieć wartości. Tu poznasz operatory zakresu i pętle. Operatory zakresu umożliwiają reprezentowanie zakresu wartości przez określenie wartości początkowej i końcowej zakresu, a dowiesz się o różnych typach operatorów zakresu. Pętle umożliwiają powtarzanie instrukcji lub sekwencji instrukcji w kółko. Możesz powtórzyć sekwencję ustaloną liczbę razy lub powtarzać sekwencję, dopóki warunek nie zostanie spełniony. Dowiesz się o różnych typach pętli używanych do tego. Pod koniec tego rozdziału dowiesz się, jak używać zakresów oraz jak tworzyć i używać różnych typów pętli (for-in, while i repeat-while). Omówione zostaną następujące tematy:

• Eksplorowanie operatorów zasięgu

• Odkrywanie pętli

Eksplorowanie operatorów zasięgu

Operatory zakresu umożliwiają reprezentowanie zakresu wartości. Załóżmy, że chcesz przedstawić sekwencję liczb rozpoczynającą się od firstNumber i kończącą lastNumber. Nie musisz określać każdej wartości; możesz po prostu określić zakres w ten sposób:

firstNmber... lastNumber

Wyobraź sobie, że musisz napisać program dla domu towarowego, który automatycznie wysyła kupon rabatowy do klientów w wieku od 18 do 30 lat. Byłoby bardzo kłopotliwe, gdybyś musiał ustawić oświadczenie if lub switch dla każdego wieku. W tym przypadku znacznie wygodniej jest użyć operatora zasięgu.

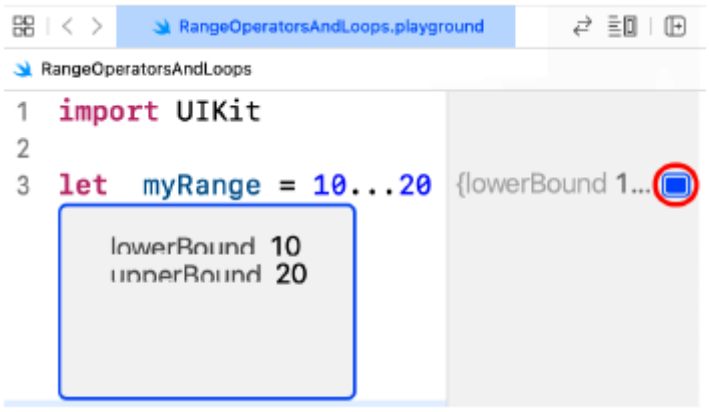
Wypróbujmy to na placu zabaw. Wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

let myRange = 10...20

Spowoduje to przypisanie sekwencji liczb, która zaczyna się od 10 i kończy na 20, włączając obie liczby, do stałej myRange. Nazywa się to operatorem zamkniętego zakresu.

2. Wynik wyświetlany w obszarze Wyniki może zostać skrócony. Kliknij kwadratową ikonę po prawej stronie wyniku. Zostanie wyświetlony w wierszu w obszarze edytora:



Możesz teraz zobaczyć pełny wynik w polu pod wierszem kodu. Jeśli chcesz, możesz przeciągnąć prawą krawędź, aby powiększyć pudełko.

Wskazówka : pamiętaj, że możesz przeciągnąć granicę między obszarami Wyniki i Edytor, aby zwiększyć rozmiar obszaru Wyniki.

3. Jeśli nie chcesz uwzględniać ostatniego numeru sekwencji w zakresie, użyj ..< zamiast ... . Wpisz i uruchom następujący kod:

let myRange2 = 10..<20

Spowoduje to zapisanie sekwencji rozpoczynającej się od 10 i kończącej się na 19 w stałej myRange2 i jest znane jako operator półotwartego zakresu.

Jest jeszcze jeden typ operatora zasięgu, jednostronny operator zasięgu, o czym dowiesz się w następnym rozdziale. Teraz, gdy już wiesz, jak tworzyć i używać zakresów, w następnej sekcji dowiesz się o pętlach, różnych typach pętli i sposobach ich używania.

Eksploracja pętli

W programowaniu często musisz robić to samo w kółko. Na przykład co miesiąc firma będzie musiała generować paski płacowe dla każdego pracownika. Jeśli firma ma 10 000 pracowników, nieefektywne byłoby napisanie 10 000 instrukcji tworzenia pasków płacowych. Lepiej byłoby powtórzyć pojedynczą instrukcję 10 000 razy, do czego służą pętle. Istnieją trzy rodzaje pętli; pętla for-in, pętla while i pętla repeatwhile. Pętla for-in będzie się powtarzać znaną liczbę razy, a pętle while i repeat-while będą powtarzać się, dopóki warunek pętli będzie spełniony. Przyjrzyjmy się po kolei każdemu typowi, zaczynając od pętli for-in, która jest używana, gdy wiesz, ile razy pętla powinna zostać powtórzona.

Korzystanie z pętli for-in

Pętla for-in przechodzi przez każdą wartość w sekwencji, a zestaw instrukcji w nawiasach klamrowych, znany jako treść pętli, jest wykonywany za każdym razem. Każda wartość jest po kolei przypisana do zmiennej tymczasowej, a zmienna tymczasowa może być używana w treści pętli. Oto jak to wygląda:

for item in sequence {

code

}

Liczba powtórzeń pętli jest podyktowana liczbą elementów w sekwencji. Zacznijmy od utworzenia pętli for-in, która wyświetli wszystkie liczby w myRange. Wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

for number in myRange {

print(number)

}

Powinieneś zobaczyć każdą liczbę w sekwencji wyświetlanej w obszarze Debug. Zauważ, że instrukcje wewnątrz pętli są wykonywane 11 razy, ponieważ myRange zawiera ostatnią liczbę z zakresu.

2. Wypróbujmy ten sam program, ale tym razem z myRange2. Zmodyfikuj kod w następujący sposób i uruchom go:

```
for number in myRange2 {

print(number)

}
```

Instrukcje wewnątrz pętli są wykonywane 10 razy, a ostatnią wartością wydrukowaną w obszarze Debug jest 19.

3. Możesz nawet użyć operatora zakresu bezpośrednio po słowie kluczowym in. Wpisz i uruchom następujący kod:

```
for number in 0...5 {

print(number)

}
```

Każda liczba od 0 do 5 jest wyświetlana w obszarze Debug.

4. Jeśli chcesz, aby sekwencja została odwrócona, użyj funkcji reversed(). Zmodyfikuj kod w następujący sposób i uruchom go:

```
for number in (0...5).reversed() {

print(number)

}
```

Każda liczba od 5 do 0 jest wyświetlana w obszarze Debug. Dobra robota! W następnej sekcji sprawdźmy pętle while, które są używane, gdy sekwencja pętli powinna być powtarzana, o ile warunek jest spełniony.

Korzystanie z pętli while

Pętla while zawiera warunek i zestaw instrukcji w nawiasach klamrowych, znany jako treść pętli. Warunek jest sprawdzany jako pierwszy; jeśli prawda, wykonywana jest treść pętli, a pętla powtarza się, dopóki warunek nie będzie fałszywy. Oto jak to wygląda:

```
while condition == true {

code

}
```

Dodaj następujący kod, aby utworzyć zmienną, zwiększ ją o 5 i kontynuuj, dopóki wartość zmiennej jest mniejsza niż 50. Kliknij przycisk Odtwórz/Zatrzymaj, aby ją uruchomić:

```
var y = 0

while y < 50 {

y += 5

print("y is \(y)")

}
```

Przejdźmy przez kod. Początkowo y jest ustawione na 0. Warunek y < 50 jest sprawdzany i zwraca prawdę, więc wykonywana jest treść pętli. Wartość y jest zwiększana o 5, a y wynosi 5 jest drukowane w obszarze Debug. Pętla się powtarza, a y < 50 jest ponownie sprawdzane. Ponieważ y wynosi teraz 5, a 5 < 50 nadal zwraca prawdę, treść pętli jest wykonywana ponownie. Powtarza się to, aż wartość y wyniesie 50, w którym to punkcie y < 50 zwraca fałsz i pętla się zatrzymuje. Jeśli warunek pętli while jest początkowo fałszywy, treść pętli nigdy nie zostanie wykonana. Spróbuj zmienić wartość y na 100, aby to zobaczyć. W następnej sekcji nauczysz się powtarzania pętli. Wykonają one najpierw instrukcje w ciele pętli przed sprawdzeniem warunku pętli.

Pętla repeat-while

Podobnie jak pętla while, pętla repeat-while zawiera również warunek i treść pętli, ale treść pętli jest wykonywana jako pierwsza przed sprawdzeniem warunku. Jeśli warunek jest prawdziwy, pętla powtarza się, dopóki warunek nie zwróci wartości false. Oto jak to wygląda:

```
repeat {

code

} while condition == true
```

Dodaj następujący kod, aby utworzyć zmienną, zwiększ ją o 5 i kontynuuj, dopóki wartość zmiennej jest mniejsza niż 50. Kliknij przycisk Odtwórz/Zatrzymaj, aby ją uruchomić:

```
var x = 0

repeat {

x += 5

print("x is \(x)")

} while x < 50
```

Przejdźmy przez kod. Początkowo x jest ustawione na 0. Wykonywana jest treść pętli. Wartość x jest zwiększana o 5, więc teraz x zawiera 5, a x wynosi 5 jest wypisywane w obszarze Debug. Warunek x < 50 jest sprawdzany, a ponieważ zwraca prawdę, pętla jest powtarzana. Wartość x jest zwiększana o 5, więc teraz x zawiera 10, a x wynosi 10 jest wypisywane w obszarze Debug. Pętla jest powtarzana, aż x zawiera 50, w którym to punkcie x < 50 zwraca fałsz i pętla się zatrzymuje. Treść pętli zostanie wykonana co najmniej raz, nawet jeśli warunek jest początkowo fałszywy. Spróbuj zmienić wartość x na 100, aby to zobaczyć. Wiesz już, jak tworzyć i używać różnych typów pętli. Niesamowite!