Proste wartości i typy

Teraz, po krótkiej prezentacji Xcode, spójrzmy na język programowania Swift. Najpierw poznasz place zabaw Swift, interaktywne środowisko, w którym możesz wpisać kod Swift i natychmiast wyświetlić wyniki. Następnie dowiesz się, jak Swift reprezentuje i przechowuje różne typy danych. Następnie przyjrzysz się kilku fajnym funkcjom języka Swift, takim jak wnioskowanie o typie i bezpieczeństwo typów, które pomogą ci pisać kod bardziej zwięźle i uniknąć typowych błędów. Na koniec dowiesz się, jak wykonywać typowe operacje na danych i jak drukować komunikaty w obszarze debugowania, aby pomóc w rozwiązywaniu problemów. Pod koniec tego rozdziału powinieneś umieć pisać proste programy, które mogą przechowywać i przetwarzać litery i cyfry.

Omówione zostaną następujące tematy:

• Zrozumienie placów zabaw Swift

• Eksplorowanie typów danych

• Odkrywanie stałych i zmiennych

• Zrozumienie wnioskowania o typie i bezpieczeństwa typu

• Eksplorowanie operatorów

• Korzystanie z instrukcji print()

Wymagania techniczne

Aby wykonać ćwiczenia będziesz potrzebować:

• Komputer Apple Mac z systemem macOS 11 Big Sur lub macOS 12 Monterey

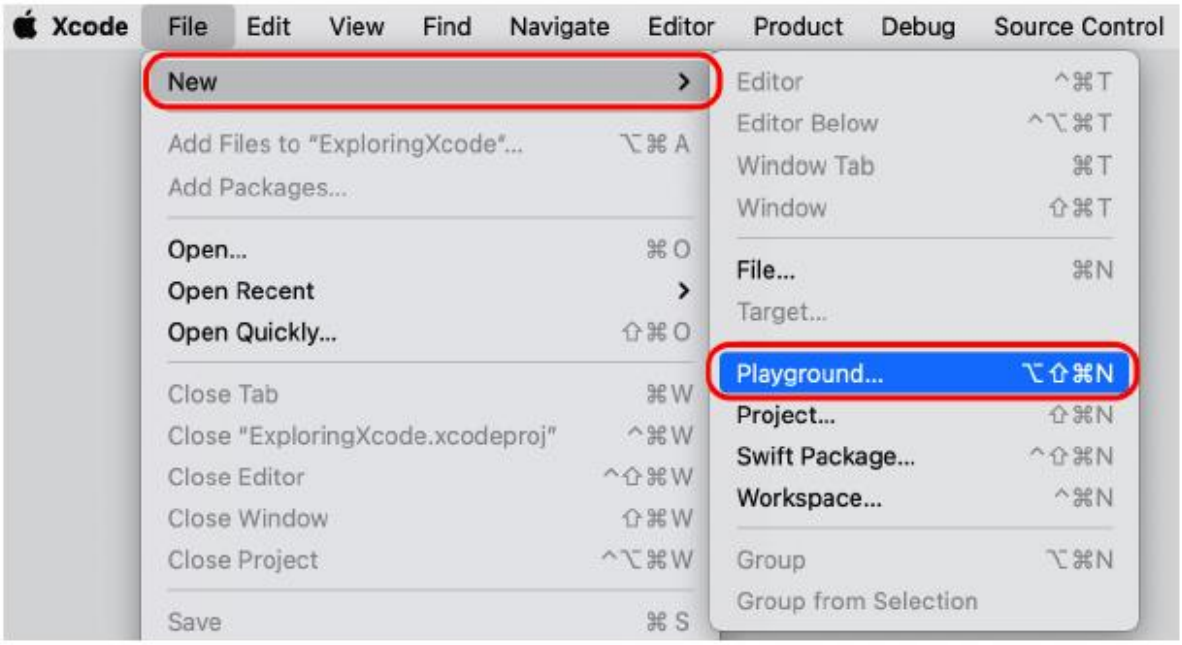
• Zainstalowany Xcode 13

Plac zabaw Xcode dla tego rozdziału znajduje się w folderze Chapter02

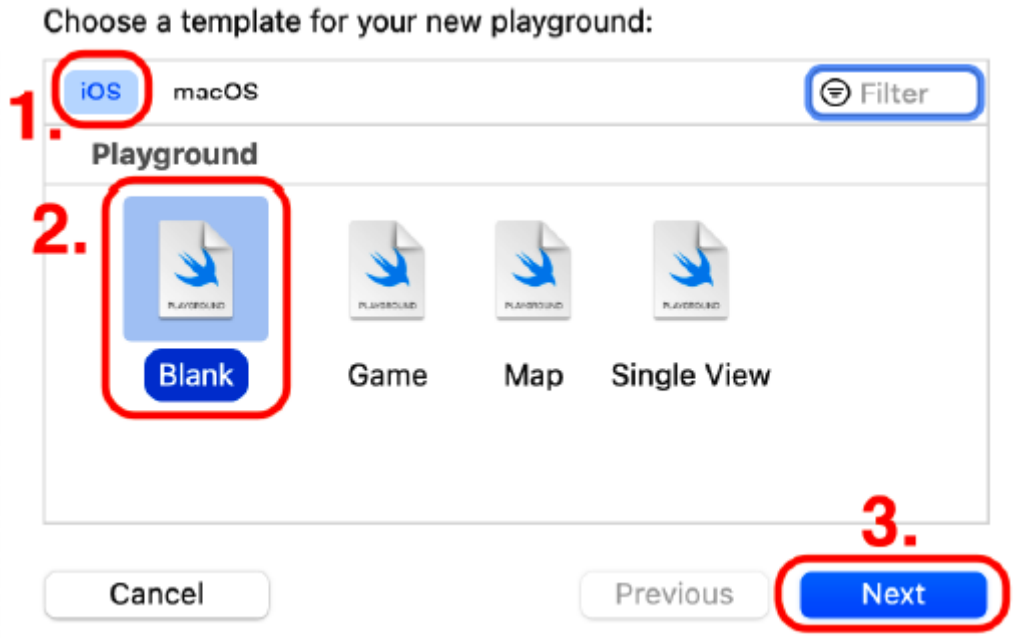
Zrozumieć place zabaw Swift

Place zabaw to interaktywne środowiska kodowania. Wpisujesz kod w lewym okienku, a wyniki są natychmiast wyświetlane w prawym okienku. To świetny sposób na eksperymentowanie z kodem i poznawanie systemowych interfejsów API. Zacznijmy od stworzenia nowego placu zabaw i zbadania jego interfejsu użytkownika. Wykonaj następujące kroki:

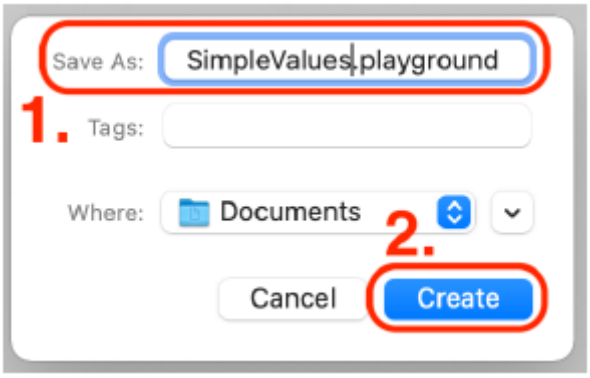
1. Aby stworzyć plac zabaw, uruchom Xcode i wybierz Plik | Nowy | Plac zabaw... z paska menu Xcode:



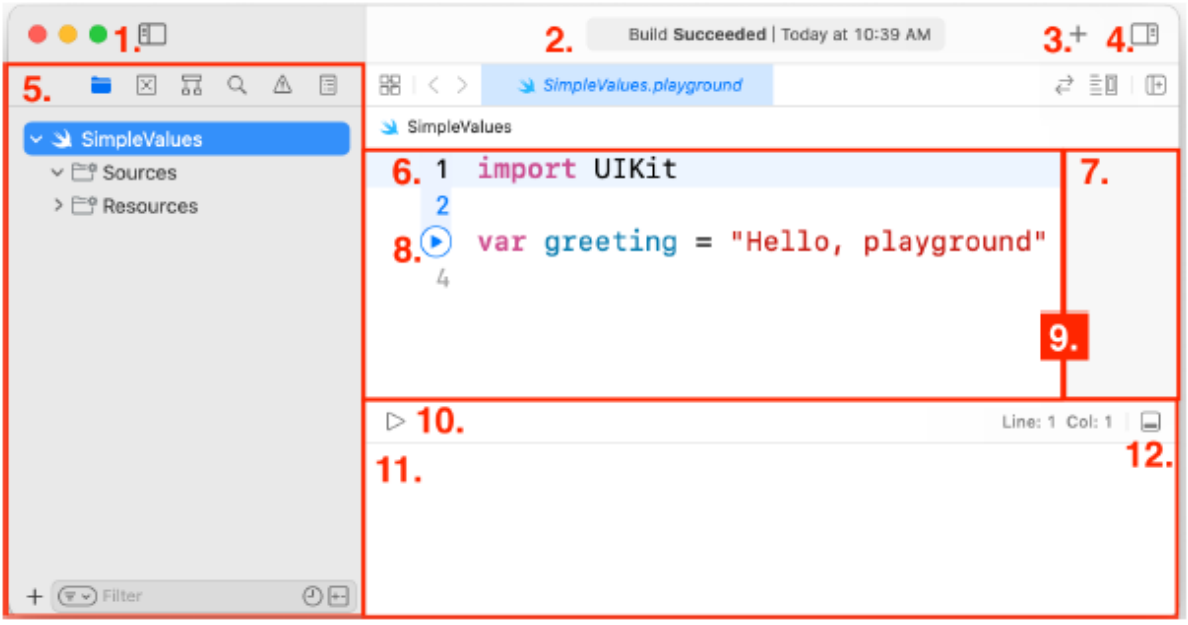
2. Pojawi się ekran szablonu. iOS powinien być już wybrany. Wybierz Puste i kliknij Dalej:



3. Nazwij swój plac zabaw SimpleValues i zapisz go w dowolnym miejscu. Po zakończeniu kliknij Utwórz:



4. Powinieneś zobaczyć plac zabaw na ekranie:



Jak widać, jest to znacznie prostsze niż projekt Xcode. Przyjrzyjmy się interfejsowi bardziej szczegółowo:

• Przycisk Nawigator (1) — Pokazuje lub ukrywa obszar Nawigatora.

• Widok aktywności (2) — pokazuje bieżącą operację lub stan.

• Przycisk Biblioteka (3) — wyświetla fragmenty kodu i inne zasoby.

• Przycisk Inspektora (4) — Pokazuje lub ukrywa obszar Inspektora.

• Obszar nawigatora (5) — zapewnia szybki dostęp do różnych części projektu. Domyślnie wyświetlany jest nawigator projektu.

• Obszar edytora (6) - Tutaj piszesz kod.

• Obszar wyników (7) — zapewnia natychmiastową informację zwrotną na temat napisanego kodu.

• Przycisk Odtwórz (8) - Wykonuje kod z wybranej linii.

• Obramowanie (9) — to obramowanie oddziela obszary Edytor i Wyniki. Jeśli okaże się, że wyniki wyświetlane w obszarze Wyniki są obcięte, przeciągnij ramkę w lewo, aby zwiększyć jej rozmiar.

• Przycisk Odtwórz/Zatrzymaj (10) – Wykonuje lub zatrzymuje wykonywanie całego kodu

na placu zabaw.

• Obszar debugowania (11) — Wyświetla wyniki polecenia print().

• Przycisk Debug (12) — Pokazuje i ukrywa obszar Debug.

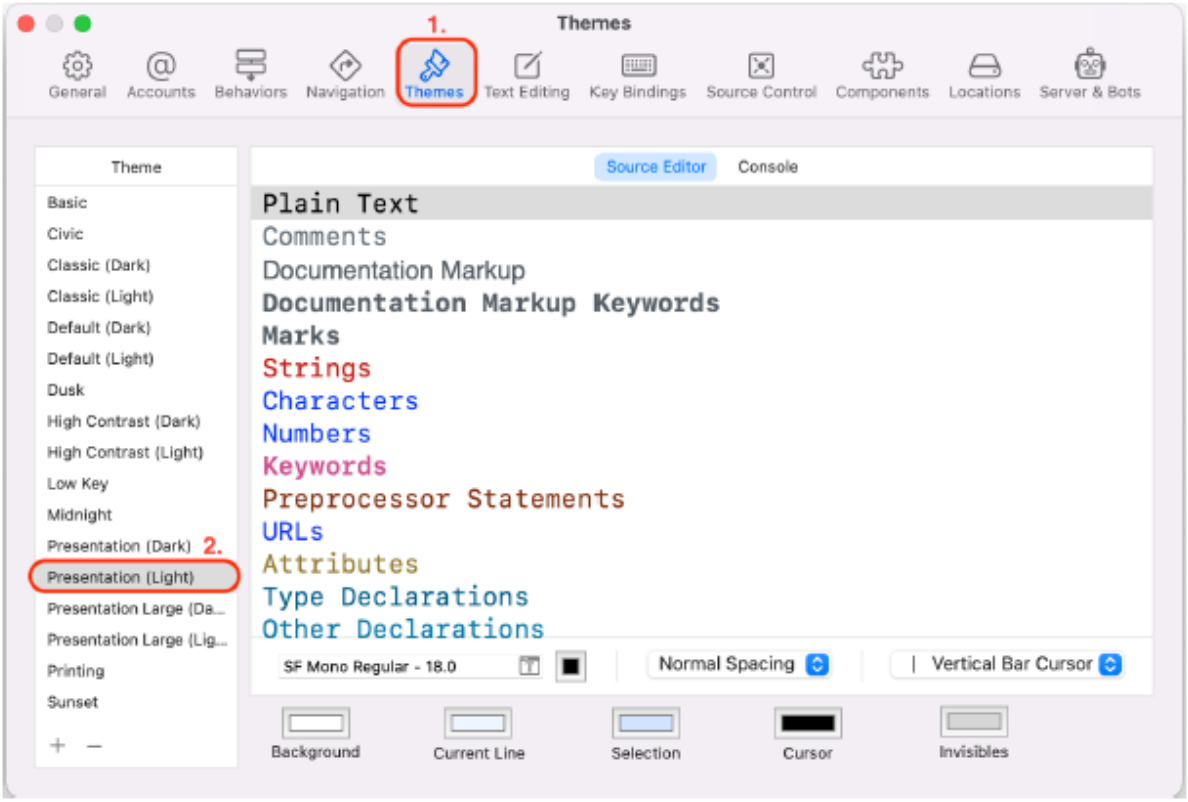
Może się okazać, że kod na placu zabaw jest zbyt mały i trudny do odczytania. Zobaczmy, jak go powiększyć w następnej sekcji.

Dostosowywanie czcionek i kolorów

Xcode ma dostępne rozbudowane opcje dostosowywania. Możesz uzyskać do nich dostęp w menu Preferencje.... Jeśli okaże się, że tekst jest mały i trudny do zauważenia, wykonaj następujące kroki:

1. Wybierz Preferencje... z menu Xcode, aby wyświetlić okno preferencji.

2. W oknie preferencji kliknij Motywy i wybierz Prezentacja (jasna), aby Twój kod był większy i łatwiejszy do odczytania:



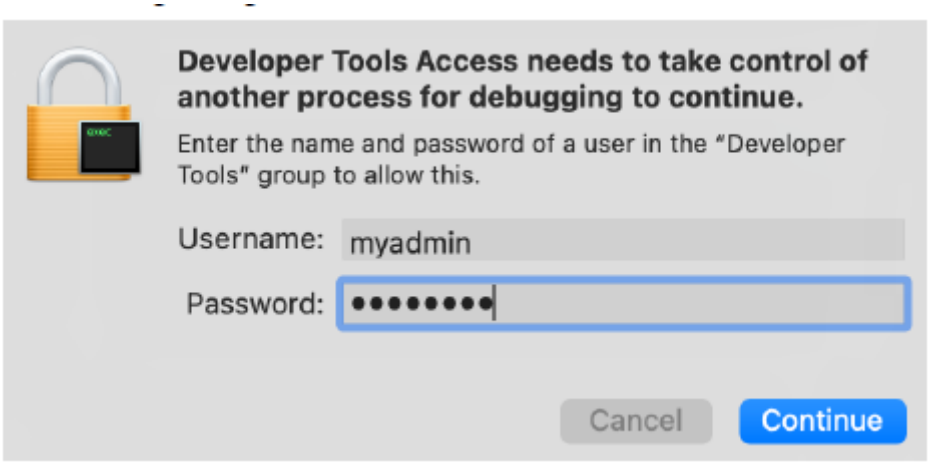
3. Zamknij okno preferencji, aby powrócić do placu zabaw. Zauważ, że tekst na placu zabaw jest większy niż wcześniej. Możesz także wypróbować inne motywy, jeśli chcesz.

Teraz, gdy dostosowałeś czcionki i kolory do swoich upodobań, zobaczmy, jak uruchomić kod na placu zabaw w następnej sekcji.

Uruchamianie kodu placu zabaw

Twój plac zabaw ma już instrukcję. Aby wykonać instrukcję, wykonaj następujące kroki:

1. Kliknij przycisk Graj/Zatrzymaj w lewym dolnym rogu placu zabaw. Możesz zobaczyć następujące okno dialogowe:



2. Wprowadź nazwę użytkownika i hasło dla konta administratora komputera Mac i kliknij przycisk Kontynuuj. Zobaczysz „Witaj, plac zabaw” wyświetlony w obszarze wyników:

Wskazówka : możesz użyć skrótu klawiszowego Command + Shift + Return, aby uruchomić kod w swoim Playground.

Aby przygotować plac zabaw do użycia w dalszej części tego rozdziału, usuń instrukcję var powitanie = "Witaj, plac zabaw" z placu zabaw. W miarę postępów wpisz kod pokazany w tym rozdziale na placu zabaw, a jeśli to konieczne, kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić. Przyjrzyjmy się prostym typom danych używanym w Swift w następnej sekcji.

Eksploracja typów danych

Wszystkie języki programowania mogą przechowywać liczby, stany logiczne i słowa, a Swift nie jest inny. Nawet jeśli jesteś doświadczonym programistą, może się okazać, że Swift reprezentuje te obiekty inaczej niż inne języki, które możesz znać.

Przejdźmy przez wersje Swift liczb całkowitych, liczb zmiennoprzecinkowych, logicznych i łańcuchów w następnych sekcjach.

Reprezentacja liczb całkowitych

Załóżmy, że chcesz zapisać:

• Liczba restauracji w mieście

• Pasażerowie w samolocie

• Pokoje w hotelu

Użyłbyś liczb całkowitych, które są liczbami bez części ułamkowej (w tym liczb ujemnych). Liczby całkowite w języku Swift są reprezentowane przez typ Int.

Reprezentowanie liczb zmiennoprzecinkowych

Załóżmy, że chcesz zapisać:

• Pi (3.14159...)

• Zero absolutne (-273.15°C)

Używałbyś liczb zmiennoprzecinkowych, które są liczbami ze składnikiem ułamkowym. Domyślnym typem liczb zmiennoprzecinkowych w języku Swift jest Double, który używa 64 bitów, w tym liczb ujemnych. Możesz także użyć Float, który używa 32 bitów, ale preferowany jest Double.

Reprezentowanie wartości logicznych

Załóżmy, że chcesz przechowywać odpowiedzi na proste pytania typu tak/nie, takie jak:

• Czy pada deszcz?

• Czy w restauracji są wolne miejsca?

W tym celu używasz wartości logicznych. Swift zapewnia typ Bool, który może być prawdą lub fałszem.

Reprezentowanie ciągów

Załóżmy, że chcesz zapisać:

• nazwę restauracji, np. „Bombay Palace”

• Opis stanowiska, taki jak „Księgowy” lub „Programista”

• Rodzaj owocu, taki jak „banan”

Należy użyć typu String firmy Swift, który reprezentuje sekwencję znaków i jest w pełni zgodny z Unicode. Ułatwia to reprezentowanie różnych czcionek i języków. Teraz, gdy wiesz, jak Swift reprezentuje te popularne typy danych, wypróbujmy je na placu zabaw, który stworzyłeś wcześniej .

Korzystanie z popularnych typów danych na placu zabaw

Wszystko, co wpiszesz na placu zabaw, zostanie wykonane, a wyniki pojawią się w obszarze Wyniki. Zobaczmy, co się stanie, gdy wpiszesz liczby, wartości logiczne i łańcuchy na swoim placu zabaw i wykonasz je. Wykonaj następujące kroki:

1. Wpisz następujący kod w obszarze edytora swojego placu zabaw:

```
// SimpleValues

42

-23

3.14159

0.1

-273.15

true
```

false

"hello, world"

"albatross"

Zauważ, że każde słowo z // przed nim jest komentarzem. Komentarze to świetny sposób na tworzenie notatek lub przypomnień dla siebie i zostaną zignorowane przez Xcode.

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby uruchomić kod.

3. Poczekaj kilka sekund. Xcode oceni wprowadzone dane i wyświetli wyniki w obszarze Wyniki w następujący sposób:

42

-23

3.14159

0.1

-273.15

true

false

"hello, world"

"albatross"

Zauważ, że komentarze nie pojawiają się w obszarze Wyniki.

Fajny! Właśnie stworzyłeś i uruchomiłeś swój pierwszy plac zabaw. Przyjrzyjmy się, jak przechowywać różne typy danych w następnej sekcji.

Odkrywanie stałych i zmiennych

Teraz, gdy znasz już proste typy danych obsługiwane przez Swift, przyjrzyjmy się, jak je przechowywać, aby móc później wykonywać na nich operacje. Do przechowywania danych można używać stałych lub zmiennych. Oba są kontenerami, które mają nazwę, ale wartość stałej można ustawić tylko raz i nie można jej zmienić po jej ustawieniu, podczas gdy wartość zmiennej można zmienić w dowolnym momencie. Musisz zadeklarować stałe i zmienne przed ich użyciem. Stałe deklarowane są za pomocą słowa kluczowego let, natomiast zmienne za pomocą słowa kluczowego var. Przyjrzyjmy się, jak działają stałe i zmienne, wdrażając je na swoim placu zabaw. Wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw, aby zadeklarować trzy stałe:

let theAnswerToTheUltimateQuestion = 42

let pi = 3.14159

let myName = "Ahmad Sahar"

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić. W każdym przypadku kontener jest tworzony i nazywany, a przypisana wartość jest przechowywana.

Wskazówka: Być może zauważyłeś, że nazwy stałych i zmiennych zaczynają się od małej litery, a jeśli w nazwie jest więcej niż jedno słowo, każde następne słowo zaczyna się od wielkiej litery. Jest to znane jako przypadek wielbłąda. Nie musisz tego robić, ale jest to zalecane, ponieważ większość doświadczonych programistów Swift przestrzega tej konwencji.

Ważna informacja : Zwróć uwagę, że sekwencja znaków ujęta w podwójny cudzysłów, „Ahmad Sahar", jest używana do przypisania wartości myName. Są one znane jako literały ciągów.

3. Dodaj następujący kod po deklaracjach stałych, aby zadeklarować trzy zmienne:

var currentTemperatureInCelsius = 27

var myAge = 50

var myLocation = "home"

Podobnie jak w przypadku stałych, kontener jest tworzony i nazwany w każdym przypadku, a przypisana wartość jest przechowywana.

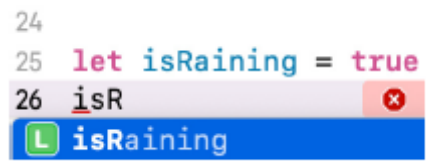
Wskazówka: Zapisane wartości są wyświetlane w obszarze Wyniki.

4. Wartość stałej nie może być zmieniona po jej ustawieniu. Aby to przetestować, dodaj następujący kod po deklaracjach zmiennych:

let isRaining = true

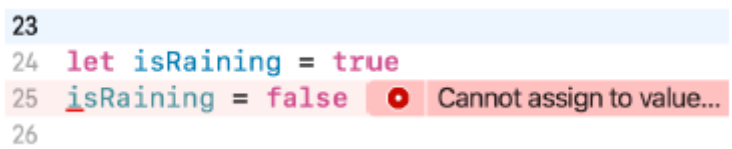
isRaining = false

Podczas wpisywania drugiego wiersza kodu pojawi się wyskakujące menu z sugestiami:



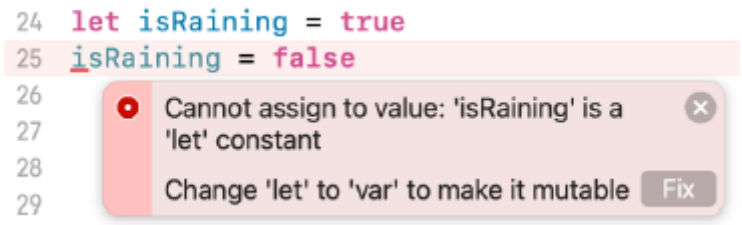
Użyj klawiszy strzałek w górę i w dół, aby wybrać stałą isRaining i naciśnij klawisz Tab, aby ją wybrać. Ta funkcja nazywa się autouzupełnianiem i pomaga zapobiegać błędom podczas wpisywania kodu.

5. Po zakończeniu pisania odczekaj kilka sekund. W drugiej linii powinieneś zobaczyć czerwone kółko z białą kropką pośrodku:



Oznacza to, że w twoim programie wystąpił błąd i Xcode myśli, że można go naprawić. Błąd pojawia się, ponieważ próbujesz przypisać nową wartość do stałej po ustawieniu jej wartości początkowej.

6. Kliknij czerwone kółko, aby rozwinąć komunikat o błędzie. Powinieneś zobaczyć następujące okno z przyciskiem Napraw:



Xcode mówi, na czym polega problem (nie można przypisać do wartości: „isRaining” jest stałą „let”) i sugeruje poprawkę (zmień „let” na „var”, aby zmienić).

7. Kliknij przycisk Napraw.

8. Powinieneś zobaczyć, że deklaracja stałej isRaining została zmieniona na deklarację zmiennej:

```
24
25 var isRaining = true
26 isRaining = false
27
```

Ponieważ nową wartość można przypisać do zmiennej po jej utworzeniu, błąd został rozwiązany. Pamiętaj jednak, że sugerowana korekta może nie być najlepszym rozwiązaniem. Jeśli spojrzysz na wpisany kod, możesz się zastanawiać, skąd Xcode rozpoznaje typ danych przechowywanych w zmiennej lub stałej. Dowiesz się, jak to się robi w następnej sekcji.

Zrozumienie wnioskowania o typie i bezpieczeństwa typów

W poprzedniej sekcji zadeklarowałeś stałe i zmienne oraz przypisałeś im wartości. Swift automatycznie określa typ stałej lub zmiennej na podstawie podanej wartości. Nazywa się to wnioskowaniem o typie. Możesz zobaczyć typ stałej lub zmiennej, przytrzymując klawisz Opcja i klikając jej nazwę. Aby zobaczyć to w akcji, wykonaj następujące kroki:

1. Dodaj następujący kod do swojego placu zabaw, aby zadeklarować ciąg:

let cuisine = "American"

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić.

3. Przytrzymaj klawisz Opcja i kliknij kuchnia, aby wyświetlić typ stałej. Powinieneś zobaczyć następujące informacje:

Jak widać, typ cuiaine to String.

Co zrobić, jeśli chcesz ustawić określony typ dla zmiennej lub stałej? Zobaczysz, jak to zrobić w następnej sekcji.

Używanie adnotacji typu do określenia typu

Widziałeś, że Xcode próbuje automatycznie określić typ danych zmiennej lub stałej na podstawie podanej wartości. Czasami jednak możesz chcieć określić typ, zamiast pozwolić, aby Xcode zrobił to za Ciebie. Aby to zrobić, wpisz dwukropek po nazwie stałej lub zmiennej, a następnie żądany typ. Nazywa się to adnotacją typu. Dodaj następujący kod do swojego placu zabaw, aby zadeklarować zmienną o określonym typie, i kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić:

var restaurantOcena: Double = 3

Tutaj określiłeś restaurantRating ma określony typ, Double. Nawet jeśli przypiszesz liczbę całkowitą do restaurantRating, będzie ona przechowywana jako liczba zmiennoprzecinkowa. W następnej sekcji dowiesz się, jak Xcode pomaga zmniejszyć liczbę błędów w programie, wymuszając bezpieczeństwo typów.

Używanie bezpieczeństwa typu do sprawdzania wartości

Swift to bezpieczny język. Sprawdza, czy przypisujesz wartości właściwego typu do zmiennych i flaguje niezgodne typy jako błędy. Zobaczmy, jak to działa, wykonując następujące kroki:

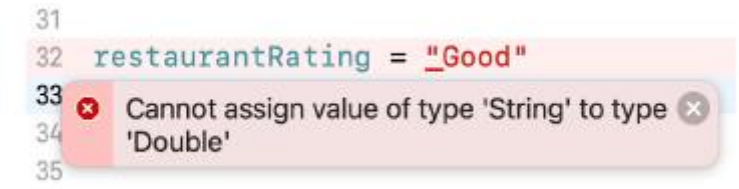
1. Dodaj następujący kod do swojego placu zabaw, aby przypisać ciąg do restaurantRating:

restaurantRating = "Good"

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby uruchomić kod.

3. Powinieneś zobaczyć czerwone kółko z x w środku. Wykrzyknik oznacza, że Xcode nie może zasugerować rozwiązania tego problemu. Kliknij czerwone kółko.

4. Ponieważ próbujesz przypisać ciąg do zmiennej typu Double, wyświetlany jest następujący komunikat o błędzie:



5. Skomentuj wiersz, wpisując // przed nim, jak pokazano:

// restaurantRating = "Good"

Czerwone kółko znika, ponieważ w twoim programie nie ma już błędów.

Wskazówka: wybranie linii kodu i wpisanie Command + / spowoduje ich skomentowanie. Teraz, gdy wiesz, jak przechowywać dane w stałych i zmiennych, przyjrzyjmy się, jak wykonywać na nich operacje w następnej sekcji.

Odkrywanie operatorów

W Swift możesz wykonywać operacje arytmetyczne, porównywania i logiczne. Operatory arytmetyczne służą do typowych operacji matematycznych. Operatory porównania i logiczne sprawdzają wartość wyrażenia i zwracają prawdę lub fałsz. Przyjrzyjmy się bardziej szczegółowo każdemu typowi operatora. Zaczniesz od operatorów arytmetycznych (dodawanie, odejmowanie, mnożenie i dzielenie) w następnej sekcji.

Korzystanie z operatorów arytmetycznych

Możesz wykonywać operacje matematyczne na liczbach całkowitych i zmiennoprzecinkowych, używając standardowych operatorów arytmetycznych przedstawionych tutaj:

+ : Dodawanie

- : Odejmowanie

* : Mnożenie

/ : Dzielenie

Zobaczmy, jak te operatory są używane. Wykonaj następujące kroki:

1. Dodaj następujący kod, aby dodać operacje arytmetyczne do swojego placu zabaw:

let suma = 23 + 20

let wynik = 32 - suma

let suma = wynik * 5

let podziel = suma / 10

2. Kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić.

Wyniki wyświetlane w obszarze Wyniki to odpowiednio 43, -11, -55 i -5. Zauważ, że 55 podzielone przez 10 daje 5 zamiast 5,5, ponieważ obie liczby są liczbami całkowitymi.

3. Operatorzy mogą pracować tylko z operandami tego samego typu. Wprowadź następujący kod i uruchom go, aby zobaczyć, co się stanie, jeśli operandy będą różnych typów:

let a = 12

let b = 12,0

let c = a + b

Otrzymasz komunikat o błędzie, Operator binarny „+” nie może być zastosowany do operandów typu „Int” i „Double”. Dzieje się tak, ponieważ a i b są różnymi typami. Zauważ, że Xcode nie może naprawić tego automatycznie, więc nie wyświetla żadnych sugestii dotyczących naprawy.

4. Aby naprawić błąd, zmodyfikuj program w następujący sposób:

let c = Double(a) + b

Double(a) pobiera wartość przechowywaną w a i tworzy z niej liczbę zmiennoprzecinkową. Oba operandy są teraz tego samego typu i teraz możesz dodać do nich wartość z b. Wartość przechowywana w c to 24,0, a 24 zostanie wyświetlone w obszarze Wyniki. Teraz, gdy już wiesz, jak używać operatorów arytmetycznych, przyjrzysz się operatorom przypisania złożonego (+=, -=, *= i /=) w następnej sekcji.

Korzystanie z operatorów przypisania złożonego

Możesz wykonać operację na wartości i przypisać wynik do zmiennej za pomocą złożonych operatorów przypisania pokazanych tutaj:

+= : Dodaj wartość i przypisz wynik do zmiennej

-= : Odejmij wartość i przypisz wynik do zmiennej

*= : Mnoży przez wartość i przypisuje wynik do zmiennej

/=: Dzieli z wartością i przypisuje wynik do zmiennej

Zobaczmy, jak te operatory są używane. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

var aa = 1

aa += 2

aa -= 1

Wyrażenie a += 2 jest skrótem dla a = a + 2, więc wartość w a wynosi teraz 1 + 2, a 3 zostanie przypisane do a. W ten sam sposób a -= 1 jest skrótem dla a = a - 1, więc wartość w a wynosi teraz 3 - 1, a 2 zostanie przypisane do a. Teraz, gdy znasz już operatory przypisania złożonego, przyjrzyjmy się operatorom porównania (==, /=, >, <, >= i <=) w następnej sekcji.

Korzystanie z operatorów porównania

Możesz porównać jedną wartość z drugą za pomocą operatorów porównania, a wynikiem będzie prawda lub fałsz. Możesz użyć następujących operatorów porównania:

== : Równe do

!= : Nie równa się

> : Większe niż

< : Mniej niż

>= : Większe lub równe

<= : Mniejszy lub równy

Zobaczmy, jak te operatory są używane. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

```
1 == 1

2 != 1

2 > 1

1 < 2

1 >= 1

2 <= 1
```

Zobaczmy, jak to działa:

• 1 == 1 zwraca prawdę, ponieważ 1 jest równe 1.

• 2 != 1 zwraca prawdę, ponieważ 2 nie jest równe 1.

• 2 > 1 zwraca prawdę, ponieważ 2 jest większe od 1.

• 1 < 2 zwraca prawdę, ponieważ 1 jest mniejsze niż 2.

• 1 >= 1 zwraca prawdę, ponieważ 1 jest większe lub równe 1.

• 2 <= 1 zwraca fałsz, ponieważ 2 jest nie mniejsze lub równe 1.

Zwrócone wartości logiczne zostaną wyświetlone w obszarze Wyniki. Co się stanie, jeśli chcesz sprawdzić więcej niż jeden warunek? W tym miejscu pojawiają się operatory logiczne (AND, OR i NOT). Przestudiujesz je w następnej sekcji.

Korzystanie z operatorów logicznych

Operatory logiczne są przydatne, gdy masz do czynienia z dwoma lub więcej warunkami. Na przykład, jeśli jesteś w sklepie spożywczym, możesz zapłacić za przedmioty, jeśli masz gotówkę lub kartę kredytową. W tym przypadku operatorem logicznym jest OR. Możesz użyć następujących operatorów logicznych:

Zobaczmy, jak te operatory są używane. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

```
1 == 1

2 != 1

2 > 1

1 < 2

1 >= 1

2 <= 1
```

Zobaczmy, jak to działa:

• 1 == 1 zwraca prawdę, ponieważ 1 jest równe 1.

• 2 != 1 zwraca prawdę, ponieważ 2 nie jest równe 1.

• 2 > 1 zwraca prawdę, ponieważ 2 jest większe od 1.

• 1 < 2 zwraca prawdę, ponieważ 1 jest mniejsze niż 2.

• 1 >= 1 zwraca prawdę, ponieważ 1 jest większe lub równe 1.

• 2 <= 1 zwraca fałsz, ponieważ 2 jest nie mniejsze lub równe 1.

Zwrócone wartości logiczne zostaną wyświetlone w obszarze Wyniki. Co się stanie, jeśli chcesz sprawdzić więcej niż jeden warunek? W tym miejscu pojawiają się operatory logiczne (AND, OR i NOT). Przestudiujesz je w następnej sekcji.

Korzystanie z operatorów logicznych

Operatory logiczne są przydatne, gdy masz do czynienia z dwoma lub więcej warunkami. Na przykład, jeśli jesteś w sklepie spożywczym, możesz zapłacić za przedmioty, jeśli masz gotówkę lub kartę kredytową. W tym przypadku operatorem logicznym jest OR. Możesz użyć następujących operatorów logicznych:

&& : Logiczne AND - zwraca prawdę tylko wtedy, gdy wszystkie warunki są spełnione

|| : Logiczne OR - zwraca prawdę, jeśli którykolwiek warunek jest spełniony

! : Logiczne NIE - zwraca przeciwną wartość logiczną

Aby zobaczyć, jak te operatory są używane, dodaj następujący kod do swojego placu zabaw i kliknij przycisk Odtwórz/Zatrzymaj, aby go uruchomić:

```
(1 == 1) && (2 == 2)

(1 == 1) && (2 != 2)

(1 == 1) || (2 == 2)

(1 == 1) || (2!= 2)

(1 != 1) || (2!= 2)

!(1 == 1)
```

Zobaczmy, jak to działa:

• (1 == 1) && (2 == 2) zwraca prawdę, ponieważ oba operandy są prawdziwe, więc prawda ORAZ prawda zwraca prawdę.

• (1 == 1) && (2 != 2) zwraca fałsz, ponieważ jeden operand jest fałszywy, więc prawda AND fałsz zwraca fałsz.

• (1 == 1) || (2 == 2) zwraca prawdę, ponieważ oba operandy są prawdziwe, więc prawda LUB prawda zwraca prawdę.

• (1 == 1) || (2 != 2) zwraca prawdę, ponieważ jeden operand jest prawdziwy, więc prawda LUB fałsz zwraca prawdę.

• (1 != 1) || (2 != 2) zwraca fałsz, ponieważ oba operandy są fałszywe, więc fałsz LUB fałsz zwraca fałsz.

• !(1 == 1) zwraca fałsz, ponieważ 1==1 jest prawdą, więc NOT true zwraca fałsz.

Zwrócone wartości logiczne zostaną wyświetlone w obszarze Wyniki. Do tej pory pracowałeś tylko z liczbami. W następnej sekcji zobaczysz, jak wykonywać operacje na słowach i zdaniach, które są przechowywane jako ciągi przy użyciu typu String firmy Swift.

Wykonywanie operacji na ciągach

Jak widzieliście wcześniej, ciąg znaków to ciąg znaków. Są one reprezentowane przez typ String i są w pełni zgodne z Unicode.

Nauczmy się kilku typowych operacji na ciągach. Wykonaj następujące kroki:

1. Możesz połączyć ze sobą dwa ciągi za pomocą operatora +. Dodaj następujący kod do swojego placu zabaw i kliknij przycisk Graj/Zatrzymaj, aby go uruchomić:

let greeting = "Good" + " Morning"

Wartości literałów ciągów „Good” i „ Morning” są połączone, a w obszarze Results wyświetlany jest komunikat „Good Morning”.

2. Możesz łączyć łańcuchy ze stałymi i zmiennymi innych typów, rzutując je jako łańcuchy. Wpisz następujący kod i uruchom go:

let rating = 3.5

var ratingResult = "The restaurant rating is " +

String(rating)

Stała oceny zawiera 3,5, wartość typu Double. Umieszczenie oceny między nawiasami String() powoduje uzyskanie wartości przechowywanej w ratingu i utworzenie na jej podstawie nowego ciągu „3.5”, który jest łączony z ciągiem w zmiennej ratingResult, zwracając ciąg „Ocena restauracji wynosi 3.5” .

3. Istnieje prostszy sposób łączenia ciągów, zwany interpolacją ciągów. Interpolacja ciągów odbywa się poprzez wpisanie nazwy stałej lub zmiennej między „\(” i „)” w ciągu. Wpisz następujący kod i uruchom go:

ratingResult = "The restaurant rating is \(rating)"

Podobnie jak w poprzednim przykładzie, wartość w rankingu jest używana do utworzenia nowego ciągu „3,5”, zwracając ciąg „Ocena restauracji wynosi 3,5”.

Do tej pory możesz zobaczyć wyniki swoich instrukcji w obszarze Wyniki. Jednak podczas pisania aplikacji przy użyciu Xcode nie będziesz mieć dostępu do obszaru wyników, który widzisz na swoim placu zabaw. Aby wyświetlić zawartość zmiennych i stałych podczas działania programu, nauczysz się, jak wydrukować je w obszarze Debug w następnej sekcji.

Korzystanie z instrukcji print()

Jak widzieliśmy w Części 1, Zapoznanie się z Xcode, projekt Xcode nie ma obszaru Results, który ma obszar gry, ale zarówno projekt, jak i obszar gry mają obszar Debug. Użycie instrukcji print() spowoduje wydrukowanie wszystkiego między nawiasami w obszarze Debug.

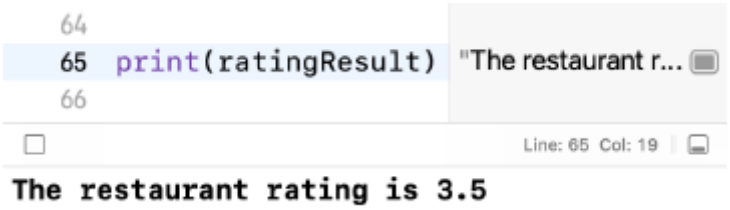
Ważna informacja

Instrukcja print() jest funkcją.

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

drukuj(ocenaWynik)

W obszarze Debug zobaczysz wartość ratingResult:



Kiedy dopiero zaczynasz, możesz użyć tylu instrukcji print(), ile chcesz. To naprawdę dobry sposób na zrozumienie, co dzieje się w Twoim programie.

Podsumowanie

W tej lekcji nauczyłeś się tworzyć i używać plików placu zabaw, które pozwalają odkrywać i eksperymentować ze Swiftem. Widziałeś, jak Swift reprezentuje różne typy danych oraz jak używać stałych i zmiennych. Umożliwia to przechowywanie w programie liczb, wartości logicznych i ciągów znaków. Poznałeś również wnioskowanie o typie, adnotację typu i bezpieczeństwo typu, co pomaga pisać kod zwięźle i z mniejszą liczbą błędów. Przyjrzałeś się, jak wykonywać operacje na liczbach i ciągach, co pozwala wykonywać proste zadania przetwarzania danych. Dowiedziałeś się, jak naprawiać błędy i jak drukować w obszarze debugowania, co jest przydatne, gdy próbujesz znaleźć i naprawić błędy w programach, które piszesz. W następnym rozdziale przyjrzysz się warunkom i opcjonalnym. Warunki warunkowe zajmują się dokonywaniem logicznych wyborów w twoim programie, a opcjonalne zajmują się przypadkami, w których zmienna może mieć wartość lub nie.