STEP IT ACADEMY

PROGRAMMING WITH python

Lekcja # 6

Pętla while

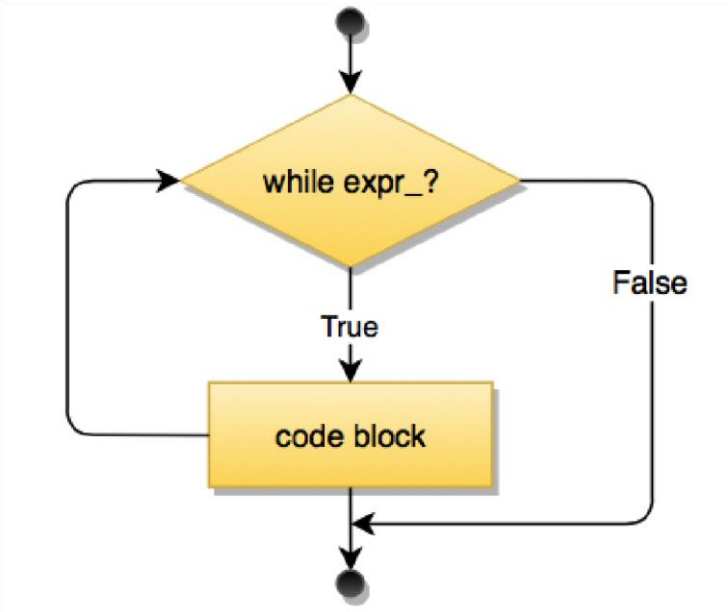
Spis treści

Pętla while

Podczas pisania programu, nie zawsze będziesz wiedział, ile konkretnie razy chcesz powtórzyć blok komend. Stworzono do tego specjalne pętle: **pętle warunkowe**.

Wyobraź sobie, że postać z gry komputerowej stanęła na lawie i traci 5 punktów życia z każdą sekundą. Będzie to trwało, dopóki gracz nie zmieni swojej pozycji.

Twórcy gry nie są w stanie przewidzieć ile będzie tych iteracji dlatego używają pętli `while` do powtórzeń (Rysunek 1).



Rysunek 1

Pętla `while` jest pętlą warunkową. Różni się od pętli `for` tym, że nie wiadomo ile razy się powtórzy.
Aczkolwiek użycie `while` jest stosunkowo podobne do `for`. Początkowo jest sprawdzany warunek; jeśli jest prawdziwy, kod należący do bloku pętli jest wykonywany, a następnie warunek jest sprawdzany ponownie. Jeśli w którymś momencie warunek przestaje być spełniony, pętla kończy swoje działanie.

Składnia pętli `while` wygląda w następujący sposób:

```
while wyrażenie_warunkowe:
   komendy
```

Spójrz na przykład użycia pętli w kodzie:

```
   # Przykład użycia pętli while
a = 5
while a<10:
   print("Now the value of a: " + str(a)+". But
         it is less than 10!")
   a = a + 1 # Increasing the value of a by 1 in
          # each iteration
print("That's it! Now the number a is greater
      than 10!")
```

Ten kod stale wyświetla wartość przechowywaną w zmiennej `a`, następnie zwiększa ją o 1. Będzie to trwało dopóki `a` jest równe lub większe 10.

Możesz użyć dowolnych warunków, prostych lub złożonych.

Pętla nieskończona

Może się zdarzyć, że będzie potrzebować powtarzać ciągle pewne czynności. **Pętla bez warunku** jest zwana **nieskończoną**.

Rozważmy **przypadek pętli nieskończonej**:

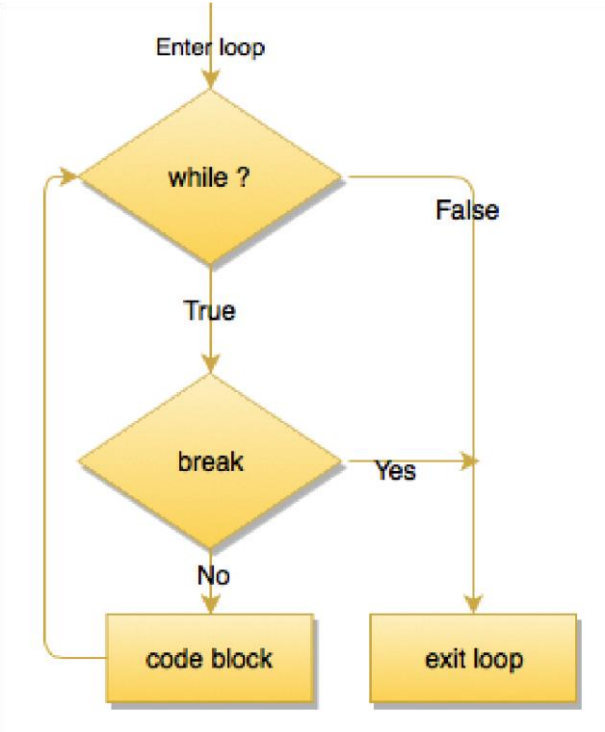
```
while True:
   instructions
```

Musisz być ostrożny podczas używania pętli nieskończonych, używaj ich tylko gdy nie masz innego wyjścia! Jeśli błędnie je utworzysz, pętla może zamrozić działanie twojego programu. W większości przypadków wystarczy dopasować odpowiedni warunek.

Instrukcja break

Instrukcja `break` **przerywa wykonywanie pętli**, na przykład jeśli pętla jest nieskończona jak w przypadku powyżej. Jest on używany w przypadku, gdy pętla musi zostać przerwana po spełnieniu odpowiedniego warunku.

Powiedzmy że chcesz pobić rekord znanego na całym świecie piłkarza Diego Maradona i odbić piłkę głową 7000 razy. Zaczniesz od stworzenia pętli `while`, a aktualny wynik będziesz zapisywał do zmiennej. Jak tylko piłka dotknie podłogi, pętla zostanie przerwana i do zmiennej nie zostanie już nic dodane (zobacz Rysunek 2 na stronie 6).



Rysunek 2

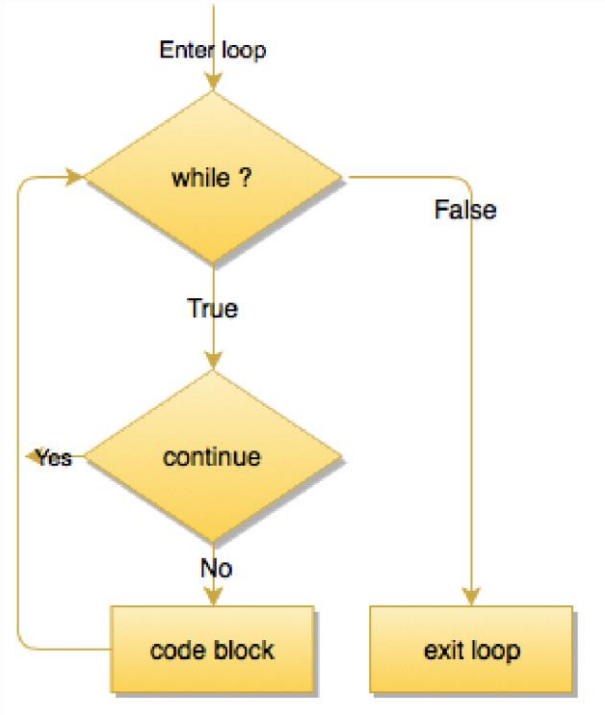
Oto w jakiś sposób operator `break` jest wykonywany w kodzie (Rysunek 3):

```
while test expression:
    # codes inside while loop
    if  condition:
        break
    # codes inside while loop

# codes outside while loop
```

Rysunek 3

Instrukcja continue

Instrukcja `continue` ignoruje kod znajdujący się za nią i przechodzi do kolejnej iteracji tak jakby kod za `continue` nie istniał (Rysunek 4).



Rysunek 4

Rysunek 5 pokazuje jak wygląda to w kodzie(see page 8). As we can see, the `continue` principle is significantly different from `break`.

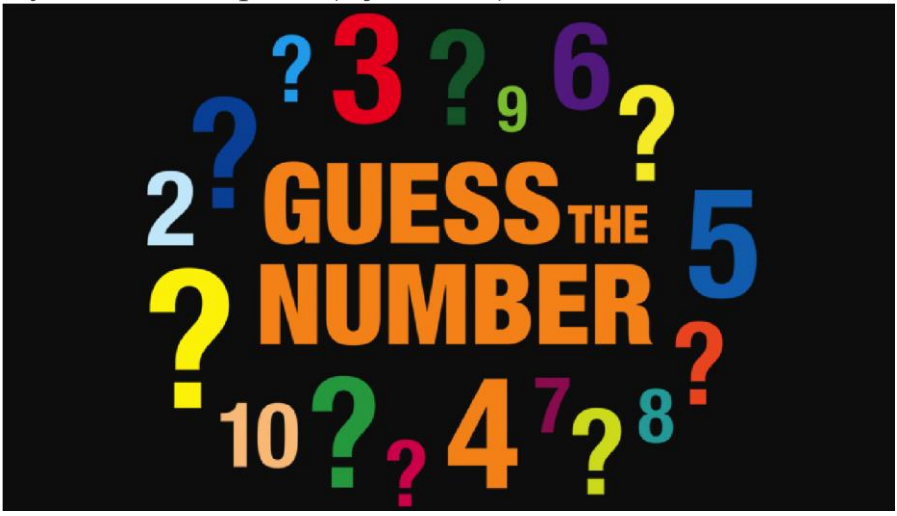
```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

Rysunek 5

Gra: zgadnij moją liczbę

Nauczyłeś się posługiwać pętlami, nadszedł czas aby zweryfikować twoją wiedzę

Stwórzmy grę **Zgadnij Moją Liczbę**. Esencja tej gry jest bardzo prosta: użytkownik musi odgadnąć cyfrę, którą wylosował komputer (Rysunek 6).



Rysunek 6

Listing 1

```
import random

print("-------Zgadnij moją liczbę-------")
print("Musisz znaleźć liczbę wybraną przez
     komputer! Liczba jest w przedziale 1 - 10")
magiczna_liczba = random.randint(1, 20)
liczba_uzytkownika = 0

while liczba_uzytkownika != magiczna_liczba:
   liczba_uzytkownika = int(input("Podaj liczbę:
                                  "))
   if magiczna_liczba > liczba_uzytkownika:
       print("Magiczna liczba jest większa!")
   elif magiczna_liczba < liczba_uzytkownika:
       print("Magiczna liczba jest mniejsza!")
print(f"Zgadłeś! Magic number:
      {magiczna_liczba}")
```

Spójrzmy jak działa nasz kod.

Dodaliśmy bibliotekę random aby generować losową liczbę. Jak pamiętasz, biblioteka powinna być dodana na początku naszego kodu.

Stworzyliśmy zmienną `magiczna_liczba` (*liczba wybrana prze komputer*) i `liczba_uzytkownika` (*liczba wybrana przez użytkownika*).

Podstawą naszej gry jest pętla:

```
while liczba_uzytkownika != magiczna_liczba:
```

Ta pętla rozpoczyna grę. Będzie ciągle pytać użytkownika o liczbę, dopóki ten nie zgadnie poprawnie:

```
while liczba_uzytkownika != magiczna_liczba:
  liczba_uzytkownika = int(input("Twoja liczba:
                                  "))
print(f"Zgadłeś! Magiczna liczba to:
      {magiczna_liczba}")
```

Stworzyliśmy bardzo prostą grę **Guess My Number**. **Oczywiście możesz ją zostawić w takim stanie ale ulepszmy ją!** Dodajmy informację jak blisko zgadnięcia był użytkownik dzięki instrukcji warunkowej `if-elif` do pętli.

Teraz możemy nie tylko sprawdzić czy gracz poprawnie zgadł liczbę ale także porównać czy magiczna liczba jest większa czy mniejsza od wprowadzonej:

```
while liczba_uzytkownika != magiczna_liczba:
   liczba_uzytkownika = int(input("Podaj liczbę:
                                   "))
   if magiczna_liczba > liczba_uzytkownika:
       print("Magiczna liczba jest większa!")
   elif magiczna_liczba < liczba_uzytkownika:
       print("Magiczna liczba jest mniejsza!")
print(f"Zgadłeś! Magic number:
      {magiczna_liczba}")
```

Lekcja # 6
Pętla while