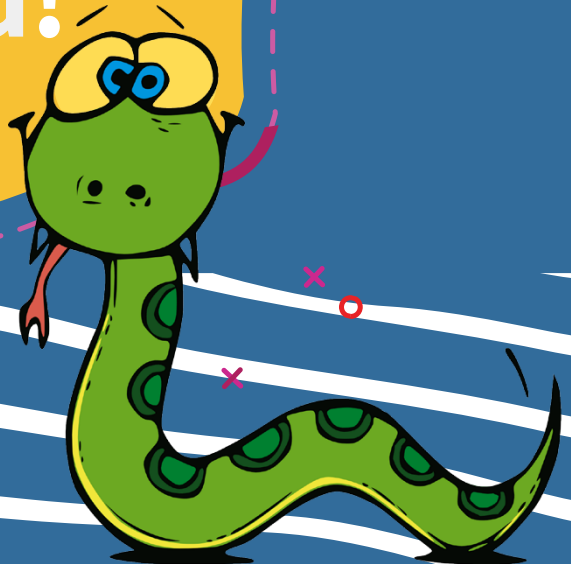


PROGRAMMING WITH python

'Hello,
world!'



Lekcja # 6

Pętla while

Spis treści

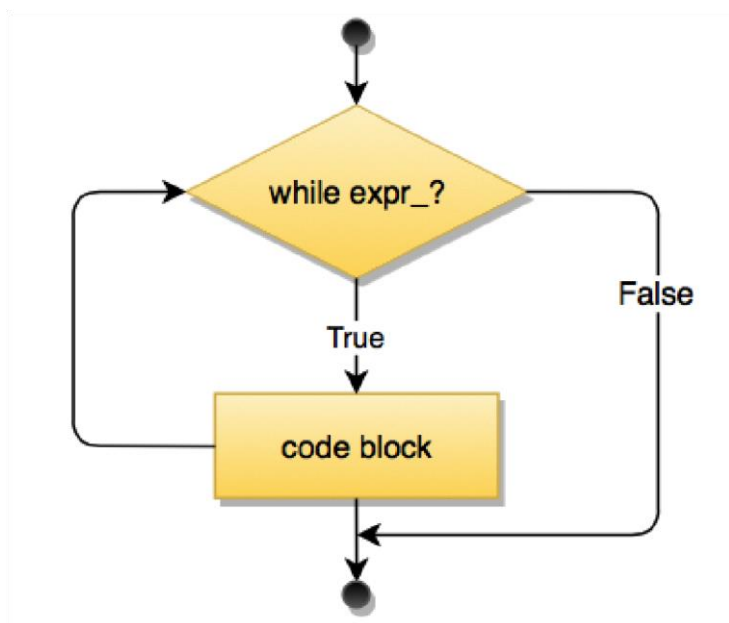
Pętla while	3
Pętla nieskończona.....	5
Instrukcja break	5
Instrukcja continue	7
Gra: zgadnij moją liczbę.....	8

Pętla while

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

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**:

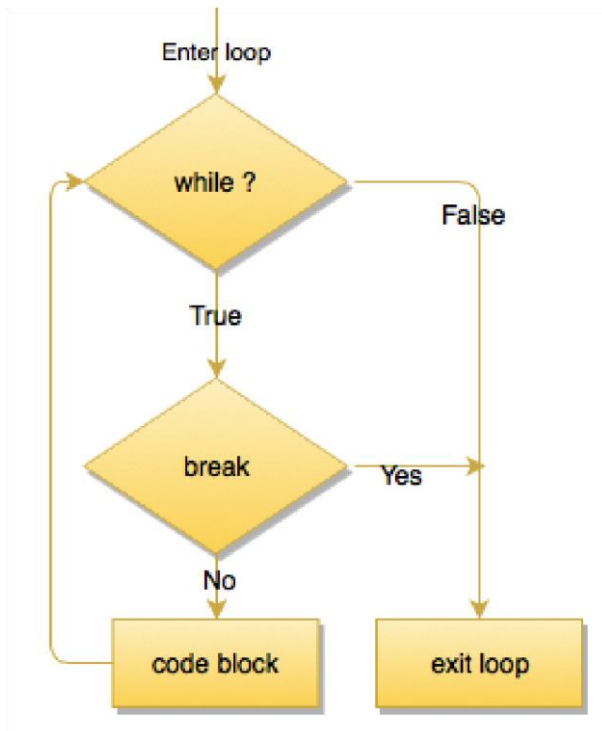
```
while True:  
    instructions
```

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. Zaczyniesz 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):

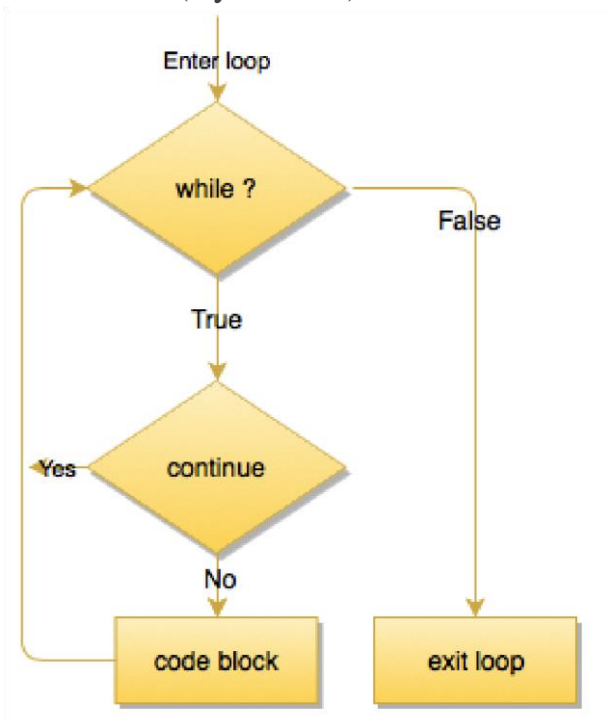
```
while test expression:
    # codes inside while loop
    if condition:
        break
    # codes inside while loop
# codes outside while loop
```

The diagram shows a code snippet for a while loop. A line from the `break` statement inside the loop is drawn and then turns into an arrow pointing to the code line immediately following the loop's closing parenthesis, representing the exit from the 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

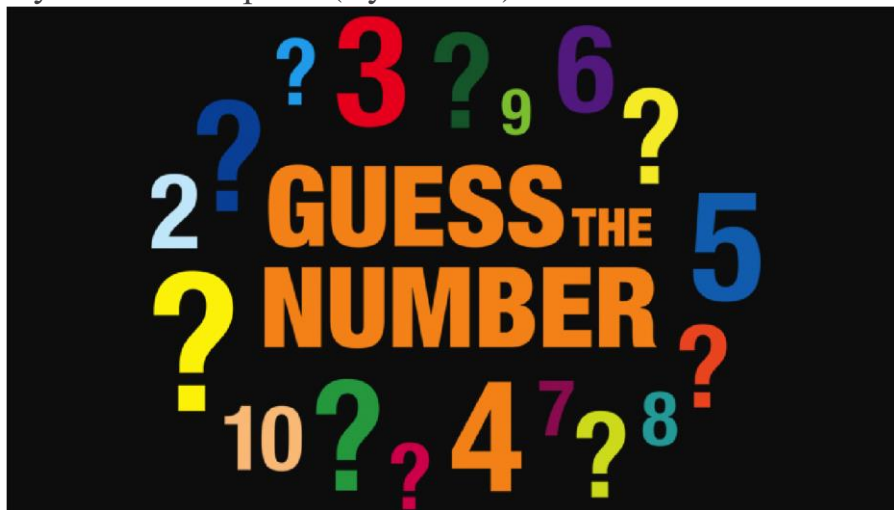
# codes outside while loop
```

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

© STEP IT Academy

www.itstep.org

Wszelkie prawa do chronionych zdjęć, audio i wideo należą do ich autorów lub prawnych właścicieli. Fragmenty prac są wykorzystywane wyłącznie w celach ilustracyjnych w zakresie uzasadnionym celem w ramach procesu edukacyjnego oraz w celach edukacyjnych zgodnie z art. 1273 ust. 4 Kodeksu cywilnego Federacji Rosyjskiej oraz art. 21 i 23 Ustawy Ukrainy "O prawie autorskim i prawach pokrewnych". Zakres i metoda cytowanych prac są zgodne z normami, nie kolidują z normalnym wykorzystaniem utworu i nie naruszają uzasadnionych interesów autorów i podmiotów praw autorskich. Cytowane fragmenty utworów można zastąpić alternatywnymi, niechronionymi analogami i jako takie odpowiadają kryteriom dozwolonego użytku. Wszelkie prawa zastrzeżone. Wszelkie powielanie, w całości lub w części, jest zabronione. Zgoda na wykorzystanie utworów i ich fragmentów jest dokonywana z autorami i innymi właścicielami praw. Materiały z tego dokumentu mogą być używane tylko z linkiem do zasobów. Odpowiedzialność za nieuprawnione kopiowanie i komercyjne wykorzystanie materiałów określa się zgodnie z obowiązującym ustawodawstwem Ukrainy.