Python #8: Pętla while

Poprzedni temat poruszał wykorzystanie pętli **FOR**. W zależności od tego ile razy chcemy wykonać daną czynność, zamykamy nasz kod w powtarzalny blok. W podobny sposób zadziała pętla **WHILE**, która wykonuje się dopóki jest spełniony zadany warunek. Tylko co to znaczy?

Tworzenie pętli w Pythonie

Przypomnijmy sobie przykład z poprzedniej części kursu:

```
name = input("Jak masz na imię? ")
print("Cześć", name)
name = input("Jak masz na imię?")
print("Cześć", name)
name = input("Jak masz na imię?")
print("Cześć", name)
```

To samo zrobimy z wykorzystaniem pętli **for**:

```
for user in range(0, 3):
    name = input("Jak masz na imię? ")
    print("Cześć", name)
```

Dla porównania możemy stworzyć warunek, aby pętla wykonywała się, dopóki licznik nie osiągnie określonej przez nas wartości:

```
counter = 0
while counter != 3:
    name = input("Jak masz na imię? ")
    print("Cześć", name)
    counter += 1
```

Kolejne wykonania pętli zwiększają nasz licznik o 1. Ponieważ zaczynamy od wartości licznika 0 wykonamy pętlę dla wartości 0, 1, 2.

Sprawdźmy to dodając wyświetlanie stanu licznika w kolejnych iteracjach:

```
counter = 0
while counter != 3:
    name = input(str(counter) + ". Jak masz na imię? ")
    print("Cześć", name)
    counter += 1
```

Stąd nasz wynik wygląda przykładowo tak:

```
0. Jak masz na imię? Maria
Cześć Maria
1. Jak masz na imię? Ania
Cześć Ania
2. Jak masz na imię? Karolina
Cześć Karolina
```

Pętla WHILE w Pythonie

W Pythonie spotykamy pętle **for** oraz **while**.
W niektórych językach programowania np. C++ mamy np. również pętlę **do while**, **w Pythonie pętla do-while nie występuje**.

WHILE rozpoczyna blok instrukcji, które będą wykonywać się tak długo, jak długo zadany warunek jest spełniony.

Warunek po każdym wykonaniu jest ponownie sprawdzany i jeśli warunek jest prawdziwy (zwraca wartość **True**), kod w bloku jest wykonywany. Jeżeli warunek jest fałszywy – przyjmie wartość **False**, blok kodu wewnątrz pętli się nie wykona. Interpreter pójdzie dalej.

```
a = 5
while a > 4:
    a = int(input("Podaj liczbę większą od 4: "))
    print("Podano liczbę a =", a)
print("Twoja liczba nie była większa od 4")
```

Załóżmy, że chcemy kupić pewnę książką, ale dopiero jak jej cena spadnie poniżej 30 zł. Pomoże nam Python, a konkretnie pętla malejąca – cenę książki będziemy regularnie obniżać.

```
book = 59
while book > 30:
    print("Czekam na przecenę")
    book = book * 0.8
    print("Przecena 20%, aktualna cena: ", round(book, 2), "zł")
    print("— — — — — — — — — — — — — — — — — —")
print("\nIdę do księgarni!")
```

Python – pętla nieskończona

Jeżeli warunek byłby cały czas spełniony, pętla wykonywałaby się bez końca.
Stąd dla pythona instrukcja while True, to po prostu pętla nieskończona.

Nie musicie jej testować, zaufajcie mi na słowo, lub przyjrzyjcie się uważnie – nie ma warunku stopu:

```
while True:
    print("Hello World!")
```

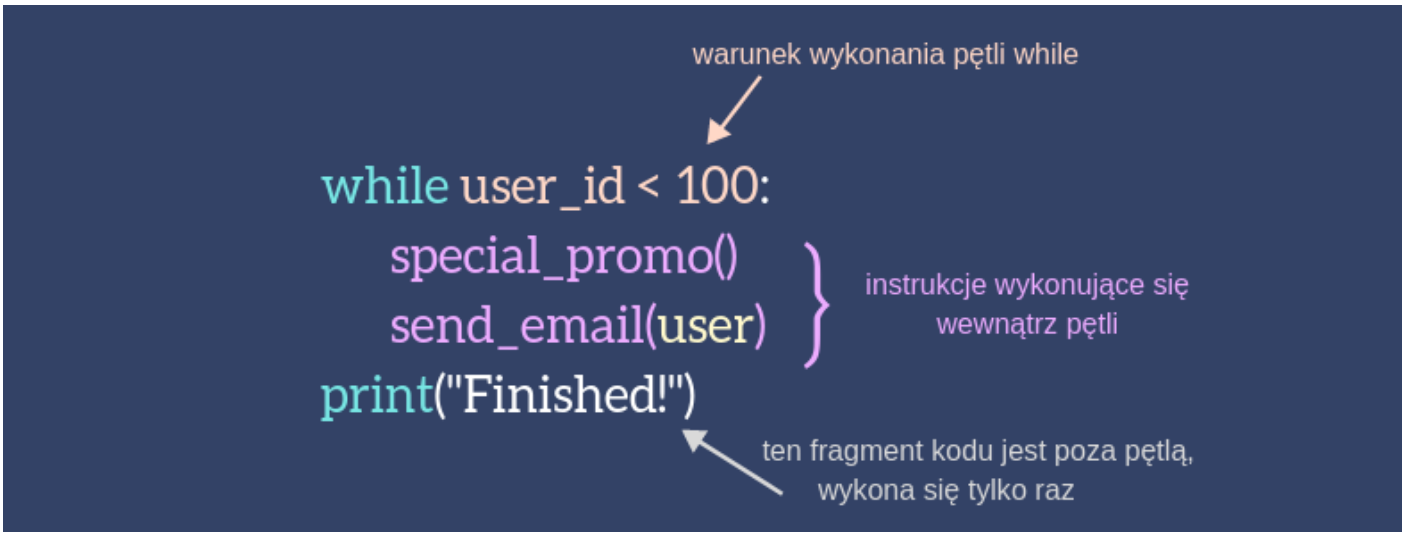
Taką samą funkcję spełni instrukcja warunkowa 1, gdyż 1 to wg Pythona wartość True.

```
while 1:
    print(bool(1))
```

Za pomocą funkcji **bool()**, wykonamy rzutowanie wartości liczbowej na wartość boolowską:

```
print(bool(1))
print(bool(0))
```

Podsumujmy co do tej pory robiliśmy korzystając z przykładu. Nasza firma się rozrosła. Na 5 lecie firmy chcemy dać specjalny kod promocyjny dla naszych pierwszych 100 klientów. Wiemy, że ich id w bazie mają numery od 0 do 99, stąd:



Pętla **while** zawiera:

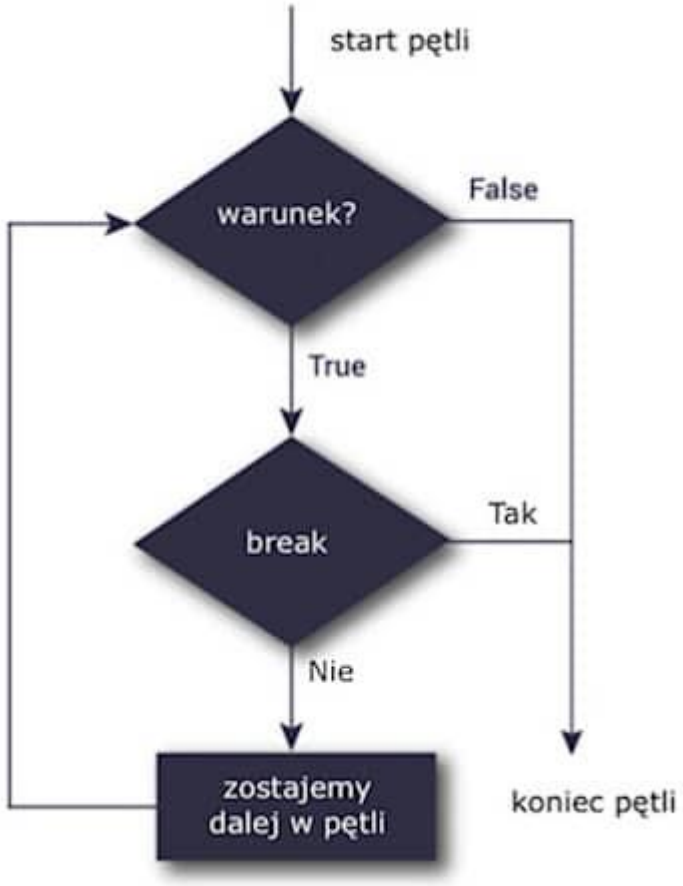
- słowo kluczowe **while**
- warunek
- blok kodu, który będzie wykonywał się w pętli (oznaczony wcięciem)

Instrukcje `break` / `continue`

Do tej pory pętla (*for* albo *while*) wykonywała się dopóki był spełniony warunek. Czasami chcemy zakończyć aktualną integrację lub nawet całą pętlę wcześniej – bez kolejnego sprawdzania wyrażenia warunkowego. Python udostępnia nam dwie instrukcje **break** oraz **continue**, które odpowiadają za natychmiastowe przerwanie bieżącej iteracji.

- **break** – kończy działanie pętli. Interpreter przechodzi do dalszej części instrukcji – następujących po bloku pętli.
- **continue** – kończy iterację bieżącej pętli. Interpreter wraca do początku pętli, wyrażenie warunkowe jest ponownie sprawdzane, aby określić, czy pętla zostanie wykonana ponownie, czy zakończyć i przejść dalej.

Instrukcja break



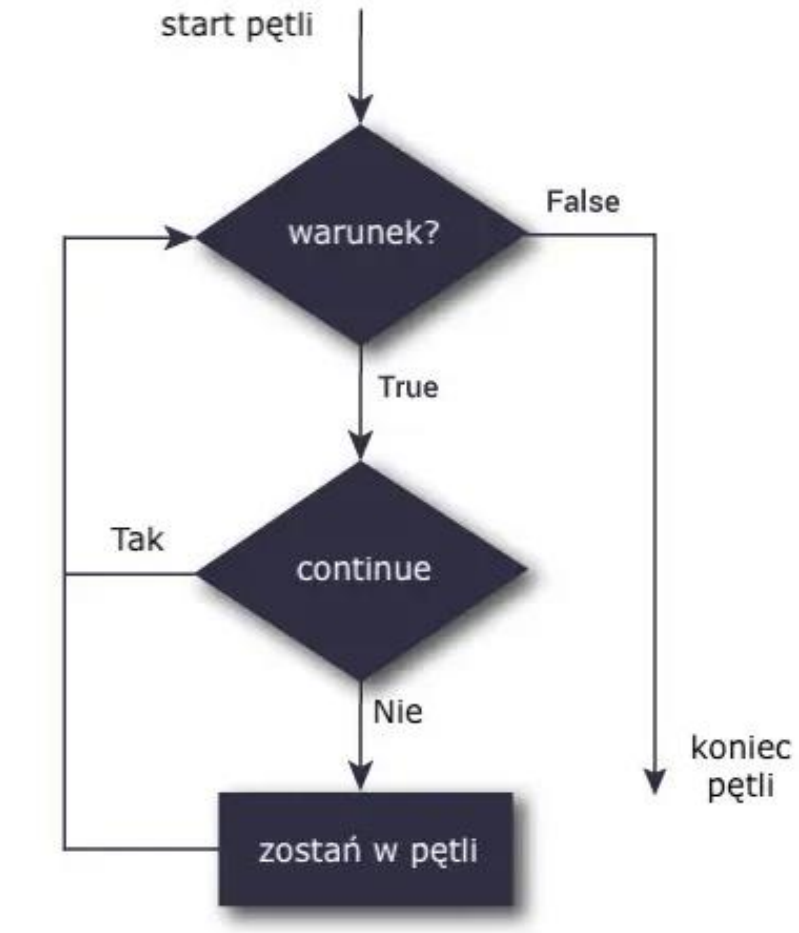
Dla zobrazowania przetestujcie poniższy kod:

```
num = 10
while num < 20:
    num += 1
    if num == 15:
        break
    print("Aktualny numer to", num)

print("Jestem poza pętlą")
```

Zobaczymy kolejno wartości 10, 11, 12, 13, 14, przy wartości 15 instrukcja **break** wymusi zakończenie wykonywania pętli i przejście do kodu poza pętlą (nie zostanie wyświetlona wartość 15). Podobnie zadziała kod z wykorzystaniem pętli **for**.

```
for num in range(10, 20):
    if num == 15:
        break
    print("Aktualny numer to", num)

print("Jestem poza pętlą")
```

Instrukcja continue



Skorzystamy z tego samego przykładu, tylko teraz zamiast **break**, użyjemy instrukcji **continue**.

```
num = 10
while num < 20:
    num += 1
    if num == 15:
        continue
    print("Aktualny numer to", num)

print("Jestem poza pętlą")
```

W tym wypadku wyświetlą się wartości od 11 do 20 bez wartości 15, przy której zostanie wywołana instrukcja continue – tzn. zostanie pominięta dalsza część pętli – część wyświetlająca „Aktualny numer to”.

```
for num in range(10, 20):
   if num == 15:
      continue
   print("Aktualny numer to", num)

print("Jestem poza pętlą")
```

Mam nadzieję, że pętle to temat dla was łatwy i przyjemny, dlatego czas na zadania!

Zadania – python pętle

Zadanie 1

Napisz program z wykorzystaniem pętli while, który dla 10 kolejnych liczb naturalnych wyświetli sumę poprzedników.
Oczekiwany wynik: 1, 3, 6, 10, 15, 21, 28, 36, 45, 55

Zadanie 2

Zapoznaj się z modułem random.

```
>>> import random
```

Stwórz prostą grę zgadywankę. Komputer losuje wartość z przedziału od 1-30. Poproś użytkownika o zgadnięcie liczby. Program pyta użytkownika o podanie liczby tak długo, dopóki gracz nie zgadnie.

Zadanie 3

Rozszerz grę z punktu powyżej. Gracz powinen otrzymać informację czy jego liczba jest za duża czy za mała.

Zadanie 4

Napisz skrypt obliczający wartość silnii. Rozwiąż zadanie za pomocą pętli for oraz pętli while.
Wejście: „Podaj dowolną liczbę całkowitą do 15:” 4
Wyjście: 4! = 24

Zadanie 5

Korzystając z modułu random stwórz kolejną prostą grę. Komputer losuje słowo z dostępnego zakresu (posiada listę słów). Następnie litery są mieszane.
Wymieszane litery pokazywane są graczowi. Gracz musi zgadnąć co to za słowo. Gracz zgaduje do skutku. Dopiero zgadnięcie przerywa grę.

Rozszerzenie: gracz może wybrać na klawiaturze „q” lub „Q”, aby zakończyć grę przed czasem.