Nazwa implementacji: Nauka języka Python– pętla for

Autor: Piotr Fiorek

Opis implementacji: Poznanie innego rodzaju pętli, jaką jest pętla for w języku Python.

Składnia pętli „for” jest następująca:

```
for <nazwa zmiennej> in <obiekt>:

instrukcja 1...

instrukcja 2...

...
```

Zamiast „<nazwa zmiennej>” wstawiamy dowolną nazwę, która będzie wykorzystywana do przechowywania kolejnych elementów pobieranych z „<obiekt>”. Najprostszym praktycznym przykładem użycia pętli „for” jest wypisanie wszystkich elementów listy:

```
>>> lista = [1, 2, 3, 4, 5]

>>> for element in lista:

... print(element)

...

1

2

3

4

5
```

W tym przykładzie pętla for pobiera kolejne elementy z listy, wrzuca je do zmiennej **„element”** i następnie przechodzi do wykonywania instrukcji, które jak zawsze są wypisane po dwukropku i we wcięciu.

Gdybyśmy chcieli z listy wyświetlić tylko parzyste elementy można by to zrobić np. tak:

```
>>> lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

>>> for elem in lista:

... if elem % 2 == 0:

... print(elem)

...

1
```

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI ORE FWIOO UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
2

4

6

8

10
```

Znak „%” jest operatorem modulo. Modulo jest to operacja wyznaczania reszty z dzielenia jednej liczby przez drugą. Czyli podczas dzielenia liczb nieparzystych przez 2, reszta z dzielenia (czyli wynik operacji modulo) jest równy 1, a dla liczb parzystych modulo jest równe 0.

Innym przykładem użycia pętli for może być sposób na szybkie wygenerowanie dużej listy, np. liczb od zera do stu lub zsumowania elementów z listy:

```
>>> lista = []

>>> for x in range(101):

... lista.append(x)

...

>>> suma = 0

>>> for x in lista:

... suma = suma + x

...

>>> print(suma)

5050
```

Najpierw tworzymy pustą listą, aby później móc do niej dodawać kolejne elementy. Kolejne elementy generujemy przy użyciu funkcji „range”. Funkcja ta przyjmuje parametr i zwraca kolejno cyfry z zakresu <0; wartość), czyli w naszym wypadku <0; 101). Kolejne liczby zwracane przez tą funkcję dowiązujemy na koniec naszej do tej pory pustej listy.

W dalszej części przykładu tworzymy zmienną równą zero, żeby móc do niej dodawać kolejne liczby. Liczby pobieramy przy użyciu pętli for do zmiennej o nazwie

„x”

i dodajemy je do zmiennej suma.

Spróbujmy stworzyć jakiś ciekawszy program wykorzystujący pętlę **„for”**. Napiszemy program, który będzie liczył znaki w naszym programie kalkulatora:

```
2
```


Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
plik = open('kalkulator.py', 'r')

suma_liter = 0

for linia in plik:

for litera in linia:

suma_liter = suma_liter + 1

print(suma_liter)

plik.close()
```

Jak widać w Pythonie, tak wydawałoby się skomplikowany program można napisać w zaledwie kilku linijkach. W pierwszej linijce używając funkcji „open” otwieramy plik. Funkcja przyjmuje dwa parametry – pierwszy to nazwa pliku, który chcemy otworzyć, więc upewnijmy się, że uruchamiamy program w tym samym katalogu, w którym znajduje się nasz plik „kalkulator.py”. Drugi parametr określa cel, w jakim chcemy otworzyć plik, my używamy „r” (od angielskiego „read”), czyli otwieramy plik tylko do czytania z niego. Można jeszcze otworzyć plik do zapisu („w” – „write”) lub tzw. aktualizacji („a” – „append”). Różnica pomiędzy dwoma ostatnimi jest taka, że przy otwarciu do aktualizacji dopisujemy na końcu pliku, a przy otwarciu do zapisu na początku i cała dotychczasowa zawartość pliku jest kasowana.

W kolejne linijce znów tworzymy zmienną, do której będziemy zliczać znaki. Następne linie to już pętle „for” – pierwsza pobiera kolejne linie z pliku, druga kolejne znaki z linii i przy każdym znaku dodaje jeden do naszej zmiennej liczącej. Na koniec wyświetlamy wynik i zamykamy plik, który wcześniej otwieraliśmy.

A gdybyśmy chcieli np. wyświetlić tylko linie zaczynające się od „if”? W Pythonie również jest to bardzo proste i posłuży nam do tego nowe słowo kluczowe „continue”. Jest ono obok „break” (którego też można używać w pętli „for”), drugim wyrażeniem służącym do sterowania pętlą. „continue” działa jednak zupełnie inaczej niż „break”. Tam gdzie „break” przerywał wykonanie pętli, „continue” przerywa tylko wykonanie jednego cyklu pętli i natychmiast rozpoczyna kolejne. Przykładowe użycie instrukcji „

continue

” może wyglądać np. tak:

```
>>> for x in range(30):

... if x % 2:

... continue

... print(x)

...

0

2

4

6

8

3
```

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI ORE FWIOO UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.

```
10

12

14

16

18

20

22

24

26

28
```

W ten sposób można wyświetlić wszystkie liczby parzyste w zakresie <0; 30). Dlaczego w „if” nie ma żadnego przyrównania? W takim przypadku jak ten zaprezentowany nie musimy porównywać wartości zwracanej przez modulo z żadną cyfrą, ponieważ zwracane wartości to albo 1 albo 0, a te cyfry same w sobie odpowiadają odpowiednio prawdzie i fałszowi. Tak naprawdę każda cyfra różna od zera jest traktowana jako prawda, a zero zawsze traktowane jest jako fałsz.

W przykładzie wiać też instrukcję „continue”. Jest ona wykonywana w momencie, kiedy modulo zmiennej „x” przez 2 wynosi jeden, czyli kiedy liczba jest nieparzysta. Wykonanie instrukcji „continue” powoduje natychmiastowe porzucenie bieżącego elementu i rozpoczęcie wykonania następnego cyklu pętli, czyli w tym przypadku pobranie następnego elementu z funkcji „range” i wykonanie instrukcji zawartych w treści pętli.

Zadania:

Napisz program, który stworzy tablicę z dziesięcioma kolejnymi cyframi (porównaj ją z analogiczną pętlą while).

Napisz program, który wypełni listę liczbami od 1 do 10, a następnie wyświetli co trzeci element tablicy.

Napisz program, który spośród liczb 1-100 wyświetli tylko te podzielne przez 7.

Napisz pętlę for, która przechodzi po wartościach od 1 do 100 i używając wyrażenia if i instrukcji continue na ekranie wyświetli tylko cyfry nieparzyste.

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
ORE
FWIOO
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Projekt “Strategia Wolnych i Otwartych Implementacji jako innowacyjny model zainteresowania kierunkami informatyczno-technicznymi oraz wspierania uczniów i uczennic w kształtowaniu kompetencji kluczowych” współfinansowany ze środków Unii Europejskiej w ramach Europejskiego Funduszu Społecznego.