Pierwsze kroki z Pythonem – dodatek

Autor: Filip Kłębczyk

Lekcja 3:

Pętle

Podczas lekcji uczniowie zapoznają się dwoma rodzajami pętli – for i while, analizując zaprezentowane przez nauczyciela przykłady. Tworzą również własne krótkie programy z ich wykorzystaniem lub modyfikują poznane przykłady. Spotykają się z pojęciem modułu w Pythonie i dowiadują się, jak z pomocą modułu random generować liczby pseudolosowe z określonych przedziałów.

Cele zajęć:

Uczeń powinien:

- Znać pojęcia: pętla, przerwanie pętli, moduł;
- Wiedzieć, jak działa pętla for i jak wykorzystywać ją w swoich programach;
- Wiedzieć, jak działa pętla while;
- Potrafi przerwać wykonywanie pętli z wykorzystaniem instrukcji break;
- Wiedzieć, jak włączyć do programu moduł z wykorzystaniem instrukcji import;
- Losować liczby z wykorzystaniem modułu random.

Pojęcia kluczowe:

Pętla, moduł

Metody pracy:

- Wykład, dyskusja, prowadzenie
- Ćwiczenia praktyczne przy komputerze
- Prezentowanie efektów pracy
- Szukanie rozwiązań na postawiony problem

Materiały pomocnicze:

- https://github.com/PyConPL/pyladies-workshop
- https://docs.python.org/

Czas na realizację zajęć: 45 min

Treści programowe (związek z podstawą programową)

Podstawa programowa z Informatyki dla szkół podstawowych, klasy VII-VIII

- I Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
  3. przedstawia sposoby reprezentowania w komputerze wartości logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów;
  5. prezentuje przykłady zastosowań informatyki w innych dziedzinach, w zakresie pojęć, obiektów oraz algorytmów.
- II Programowanie i rozwiązywanie problemów z wykorzystaniem komputera i innych urządzeń cyfrowych. Uczeń:
  1. projektuje, tworzy i testuje programy w procesie rozwiązywania problemów. W programach stosuje: instrukcje wejścia/wyjścia, wyrażenia arytmetyczne i logiczne, instrukcje warunkowe, instrukcje iteracyjne, funkcje oraz zmienne i tablice. W szczególności programuje algorytmy z działu I pkt 2;
- III Posługiwanie się komputerem, urządzeniami cyfrowymi i sieciami komputerowymi. Uczeń:
  2. rozwija umiejętności korzystania z różnych urządzeń do tworzenia elektronicznych wersji tekstów, obrazów, dźwięków, filmów i animacji;
  3. poprawnie posługuje się terminologią związaną z informatyką i technologią.
- IV Rozwijanie kompetencji społecznych. Uczeń:
  1. bierze udział w różnych formach współpracy, jak: programowanie w parach lub w zespole, realizacja projektów, uczestnictwo w zorganizowanej grupie uczących się, projektuje, tworzy i prezentuje efekty wspólnej pracy;

Przebieg zajęć:

1. Wprowadzenie – 5 min

Pytamy uczniów, czy spotkali się z pojęciem pętli w programowaniu? Jeżeli nie potrafią wytłumaczyć działania pętli lub mają z tym poważne problemy, wyjaśniamy to pojęcie oraz staramy się je przedstawić na łatwych do wyobrażenia przykładach (np. sygnalizator świetlny dla przechodniów, który w sposób ciągły powtarza tę samą sekwencję trzech rodzajów świateł – czerwone, zielone, migające zielone).

2. Część zasadnicza - 35 min

Nauczyciel prezentuje na początku działanie pętli for. W Pythonie ma ona następującą formę:

```
for i in iterowalny:
    Blok instrukcji
```

Pętla for w Pythonie jest wykorzystywana do wykonywania kolejnych kroków na tzw. iterowalnych elementach. Mogą to być niektóre typy kontenerowe (np. listy), ciągi znakowe lub specjalne rodzaje funkcji – generatory (np. range). Przykładowo, możemy wypisać w osobnych liniach wszystkie znaki wprowadzone przez użytkownika.

```
1  txt = input()
2  for i in txt:
3      print(i)
```

Zmienna i w kolejnych iteracjach pętli przyjmuje znaki z ciągu znakowego (tekstu) zapisanego w zmiennej txt.

Możemy też wykonać pętlę określoną liczbę razy z wykorzystaniem generatora range. Poniższy kod wypisze tekst wewnątrz pętli 5 razy.

```
1  for i in range(5):
2      print("Ahoj przygodo!")
```

Możemy też wypisać w kolejnych obiegach pętli zmienną i. Warto zwrócić uwagę, że przyjmuje ona pięć wartości począwszy od 0, a nie 1.

```
1  for i in range(5):
2      print(i)
```

Powyższy program wypisze pięć wartości – liczby od 0 do 4. Jeśli chcielibyśmy wypisać zamiast tego wartości od 1 do 5, jednym ze sposobów na to jest dodanie 1 przy wypisywaniu.

```
1  for i in range(5):
2      print(i+1)
```

Można też ten sam efekt osiągnąć inaczej – podając zakres (w postaci dwóch parametrów), po którym ma odbywać się iteracja, przy czym należy pamiętać, że druga z liczb z zakresu nie zostanie podstawiona pod zmienną i, więc całość zakończy się na liczbie wcześniejszej przed nią, czyli w poniższym przykładzie na 5.

```
1  for i in range(1, 6):
2      print(i)
```

Funkcja-generator range przyjmuje tutaj dwa argumenty – używamy, tak jak w każdej funkcji, przecinka do ich rozdzielenia. W końcu możemy podać jeszcze jeden argument, który określa, jak szybko w kolejnych krokach mają się zwiększać liczby w podanym zakresie. Przykładowo poniższy kod wypisze liczby 1, 3 i 5, gdyż trzeci argument dla range określa, że mają się zmieniać (z krokiem) co 2.

```
1  for i in range(1, 6, 2):
2      print(i)
```

By przećwiczyć nowe informacje, zadajemy uczniom dwa zadania. Zadanie pierwsze polega na wypisaniu wszystkich parzystych liczb dwucyfrowych (bez korzystania z warunku if i operatora %). Do rozwiązania należy wykorzystać range w wersji z trzema argumentami:

```
1  for i in range(10, 100, 2):
2      print(i)
```

Drugie zadanie to wypisanie po kolei wszystkich wielokrotności liczby od 5 do 60 (włącznie).

```
1  for i in range(5, 65, 5):
2      print(i)
```

Aby nieco urozmaicić nasze zadania, skorzystamy z modułu random. Moduły (biblioteki) w Pythonie umożliwiają korzystanie z dodatkowych funkcji i innych elementów, niedostępnych dla programisty bez ich wczytania z wykorzystaniem słowa kluczowego import. W poniższym przykładzie wczytujemy bibliotekę random, która zawiera przydatną funkcję randint losującą liczbę całkowitą z określonego przedziału (obustronnie domkniętego, czyli inaczej niż w przypadku range). Program wypisze wylosowaną liczbę z przedziału od 1 do 6 (jest to działanie analogiczne do rzutu typową kostką do gry o sześciu ścianach).

```
1  import random
2
3  print(random.randint(1,6))
```

Przykładowo, jeżeli chcielibyśmy wylosować trzy razy liczby z przedziału od 1 do 6 (analogicznie do trzech rzutów kostką do gry), możemy to zrealizować z wykorzystaniem pętli for.

```
1  import random
2
3  for i in range(3):
```

```
4     print(random.randint(1,6))
```

Możemy też o liczbę losowań zapytać użytkownika.

```
1  import random
2
3  print("Ile razy mam rzucić kostką?")
4  x = int(input())
5  print("Wyniki rzutów kostką:")
6  for i in range(x):
7      print(random.randint(1,6))
```

W Pythonie pętle możemy też przerwać szybciej, używając słowa kluczowego break. Wyobraźmy sobie sytuację, że losujemy liczbę z przedziału od 1 do 6 dziesięć razy, ale jeśli tylko wypadnie 6, to przerywamy dalsze losowanie. Odpowiadający kod przedstawiamy uczniom poniżej:

```
1  import random
2
3  print("Wyniki rzutów kostką:")
4  for i in range(10):
5      result = random.randint(1,6)
6      print(result)
7      if result == 6:
8          break
9  print("Liczba losowań to",i+1)
```

Uczniom dajemy zadanie modyfikacji powyższego programu:

a) program na końcu ma wyświetlić sumę wszystkich wylosowanych liczb;
b) program na końcu ma wyświetlić, ile razy wylosowano liczbę parzystą.

Pierwsza modyfikacja (przykładowa):

```
1  import random
2
3  count_sum = 0
```

```
print("Wyniki rzutów kostką:")
for i in range(10):
    result = random.randint(1, 6)
    count_sum = count_sum + result
    print(result)
    if result == 6:
    break
print("Liczba losowań to", i+1)
print("Suma losowań to", count_sum)
```

Druga modyfikacja (przykładowa):

```
import random

count_sum = 0
count_even = 0
print("Wyniki rzutów kostką:")
for i in range(10):
    result = random.randint(1, 6)
    count_sum = count_sum + result
    print(result)
    if result % 2 == 0:
        count_even = count_even + 1
    if result == 6:
        break
print("Liczba losowań to", i+1)
print("Suma losowań to", count_sum)
print("Liczbę parzystą wyrzucono", count_even, "razy")
```

Ostatni program możemy trochę usprawnić. Liczba 6 jest liczbą parzystą, więc możemy sprawdzać, czy została wyrzucona tylko w tym przypadku, gdy wiemy, że wyrzucono liczbę parzystą (wyrzucenie nieparzystej eliminuje liczbę 6 jako jedną z możliwości).

```
import random

count_sum = 0
count_even = 0
print("Wyniki rzutów kostką:")
```

```
6   for i in range(10):
7       result = random.randint(1, 6)
8       count_sum = count_sum + result
9       print(result)
10      if result % 2 == 0:
11          count_even = count_even + 1
12          if result == 6:
13              break
14  print("Liczba losowań to", i+1)
15  print("Suma losowań to", count_sum)
16  print("Liczbę parzystą wyrzucono", count_even, "razy")
```

Powyższy program jest o tyle lepszy, że jeżeli wylosowana liczba nie jest liczbą parzystą, to program nie musi sprawdzać, czy jest równa 6, co w przypadku bardzo wielu losowań daje pewną oszczędność czasu wykonywania programu (inaczej mówiąc, lepszą wydajność czasową programu).

W Pythonie, gdy nową wartość zmiennej opieramy o zwiększenie jej samej, możemy skorzystać z operatora +=.

W związku z tym wyrażenie:

```
count_sum = count_sum + result
```

Możemy zapisać krócej jako:

```
count_sum += result
```

Podobnie jak zwiększenie o jeden (inkrementacja):

```
count_even = count_even + 1
```

Możemy zapisać jako:

```
count_even += 1
```

Gdybyśmy natomiast zmniejszali wartość, możemy zastosować operator -=, tak jak poniżej zmniejszamy o jeden (dekrementujemy) wartość zmiennej x:

```
x -= 1
```

Istnieją też odpowiednio operatory takie jak *= /= //= %=, których działanie jest analogiczne do wcześniejszych przykładów.

Poza pętlą for w języku Python istnieje też pętla while. Jest to pętla, która powtarza instrukcje w niej zawarte w zależności, czy warunek (kontynuacji) jest spełniony. W momencie gdy warunek przestanie być prawdziwy, kolejny krok pętli nie jest wykonywany. Podobnie jak w przypadku pętli for istnieje również możliwość przerwania w dowolnym miejscu pętli z wykorzystaniem instrukcji break.

```
while warunek:
    Blok instrukcji
```

Na koniec lekcji przedstawiamy przykładowy program wykorzystujący pętle while – wypisujący wielokrotności liczby 2, ale mniejsze lub równe 128.

```
1  x = 2
2  while x <= 128:
3      print(x)
4      x *= 2
```

3. Podsumowanie – 5 min.

Powtarzamy nowe informacje, które pojawiły się w trakcie lekcji. Pytamy uczniów, jakie elementy sprawiły im największą trudność oraz zapowiadamy, że więcej przykładów z pętlą while będzie zaprezentowanych na kolejnej lekcji.