

# Pierwsze kroki z Pythonem – dodatek

Autor: Filip Kłębczyk

## Lekcja 2:

### Instrukcja warunkowa

Podczas zajęć uczniowie zapoznają się z różnymi wariantami konstrukcji instrukcji warunkowej if oraz dowiadują się, jak porównywać dwie wartości z wykorzystaniem odpowiednich operatorów. Nabytą wiedzę utrwalają poprzez pisanie programów rozwiązujących postawione problemy.

#### Cele zajęć:

Uczeń powinien:

- Znać pojęcia: instrukcja warunkowa;
- Wiedzieć, jak wykorzystać konstrukcje warunkowe do rozpatrywania różnych przypadków;
- Wiedzieć, jakie operatory służą do porównywania wartości i potrafić stosować je w swoich programach;
- Prawidłowo wykorzystywać wcięcia do oznaczania bloków kodu w ramach instrukcji warunkowej.

#### Pojęcia kluczowe:

Instrukcja warunkowa, wcięcie, wcięty blok kodu

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;

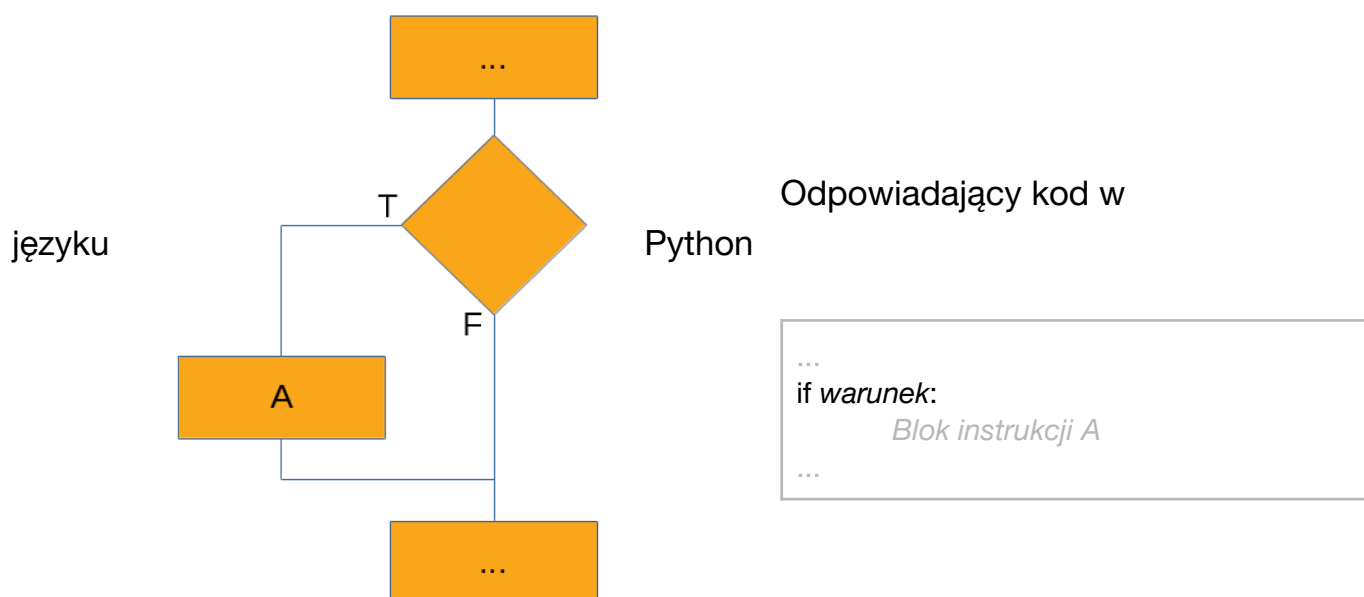
# Przebieg zajęć:

## 1. Wprowadzenie – 5 min

Przypominamy uczniom krótko pojęcia z ostatnich zajęć. Informujemy, że w ramach bieżącej lekcji nauczą się stosować instrukcję warunkową `if` w języku Python. Pytamy, czy spotkali się z tym pojęciem w programowaniu. Jeżeli uczniowie nie potrafią wytłumaczyć działania instrukcji warunkowej lub mają z tym poważne problemy, to wyjaśniamy pojęcie oraz staramy się je zobrazować na łatwo przystępnych przykładach (np. zwrotnica kolejowa decydująca o tym, którym torem pojedzie dalej pociąg).

## 2. Część zasadnicza – 35 min

Zajęcia zaczynamy o przedstawienia najprostszej postaci instrukcji warunkowej w Pythonie i odpowiadającego jej schematu blokowego.



Tłumaczymy działanie – w momencie gdy warunek (wyrażenie logiczne) jest spełniony to wykonuje się pewien zestaw operacji (na schemacie oznaczony jako A). W języku Python taki zestaw zawiera wcięcie (zwykle zalecane wcięcie to 4 spacje, edytor Mu dodaje je po wciśnięciu klawisza TAB) w każdej linii w stosunku do linii, gdzie pojawiało się słowo kluczowe `if` i

warunek. Bardzo częstym błędem wśród początkujących jest zapominanie o konieczności pojawienia się znaku `:` w linii z warunkiem lub wcięcia kodu wykonującego się po spełnieniu warunku. Zastosowanie instrukcji warunkowej przedstawiamy na prostym przykładzie – piszemy program, który podnosi liczbę podaną przez użytkownika do potęgi, dodatkowo przed wyświetleniem wyniku sprawdza, czy liczba jest podzielna przez 3 i informuje o tym na ekranie.

```
1 print("Podaj liczbę aby uzyskać jej drugą
2 potęgę:")
3 x = int(input())
4 if x % 3 == 0:
5     print("Liczba jest podzielna przez 3")
   print("Drugą potęgą tej liczby jest", x**2)
```

W programie warunkiem jest wyrażenie `x % 3 == 0` (3 linia kodu). Najpierw wykonywana jest operacja `x % 3`, czyli obliczenie reszty z dzielenia przez 3, a następnie z wykorzystaniem operatora `==` sprawdzenie, czy jest ona równa 0. Jeśli warunek jest spełniony (brak reszty), to wypisywany jest tekst "Liczba jest podzielna przez 3" (linia 4). Natomiast zawsze wypisana zostanie druga potęga (linia 5 kodu), ponieważ drugie wywołanie funkcji `print` znajduje się w linii kodu, która nie jest wcięta, tak jak poprzednia linia, a zatem jej wykonanie nie zależy od warunku. Jest to ważna cecha Pythona, gdyż w większości innych języków programowania blok kodu wykonywany po spełnieniu warunku, w ramach pętli lub wewnątrz funkcji/klasz jest oznaczany nawiasami klamrowymi lub słowami kluczowymi (np. `begin-end`). W Pythonie natomiast do tego celu wykorzystujemy wcięcia poprzedzone dwukropkiem w linii przed wcięciem. Przykładowo możemy sobie wyobrazić kilka poziomów wcięć:

```
Blok 1:
  Blok 2:
    Blok 3
  Blok 2
Blok 1
```

Przy porównaniach dosyć częstym błędem początkujących jest mylenie operatora równości `==` z operatorem przypisania `=`. Pierwszy służy do

sprawdzenia, czy mamy takie same (równe) wartości po obu jego stronach, drugi do przypisania (ustawienia) wartości zmiennej po jego lewej stronie.

Poniżej sprawdzamy, czy wartość zmiennej x jest równa 2.

```
x == 2
```

Poniżej przypisujemy do x wartość 2.

```
x = 2
```

Liczby możemy też porównywać z wykorzystaniem różnych operatorów, które przedstawiamy uczniom:

- > (większy)
- >= (większy lub równy)
- < (mniejszy)
- <= (mniejszy lub równy)
- == (równy)
- != (różny)

Następnie prosimy uczniów o rozbudowanie programu tak, aby poza podzielnością przez 3 sprawdzał dodatkowo też

- podzielność przez 4 i informował o tym fakcie;
- czy podana liczba jest równa 13 i w takim przypadku wypisywał “Ta liczba uważana jest za pechową”.

Przykładowe rozwiązanie będzie wyglądało następująco:

```
1 print("Podaj liczbę aby uzyskać jej drugą potęgę:")
2 x = int(input())
3 if x % 3 == 0:
4     print("Liczba jest podzielna przez 3")
5 if x % 4 == 0:
6     print("Liczba jest podzielna przez 4")
7 if x == 13:
8     print("Podana liczba jest uważana za pechową")
```

```
9 print("Drugą potęgą tej liczby jest", x**2)
```

Następnym zadaniem dla uczniów jest takie rozbudowanie programu, by w przypadku, gdy jakaś liczba jest podzielna przez 3, podawał w kolejnej linii wynik takiego dzielenia. Chcemy analogicznego działania programu również przy sprawdzaniu podzielności przez 4. Na przykład dla liczby 24 chcemy zobaczyć następujące komunikaty na ekranie:

```
Liczba jest podzielna przez 3
Wynik takiego dzielenia to 8
Liczba jest podzielna przez 4
Wynik takiego dzielenia to 6
Drugą potęgą tej liczby jest 576
```

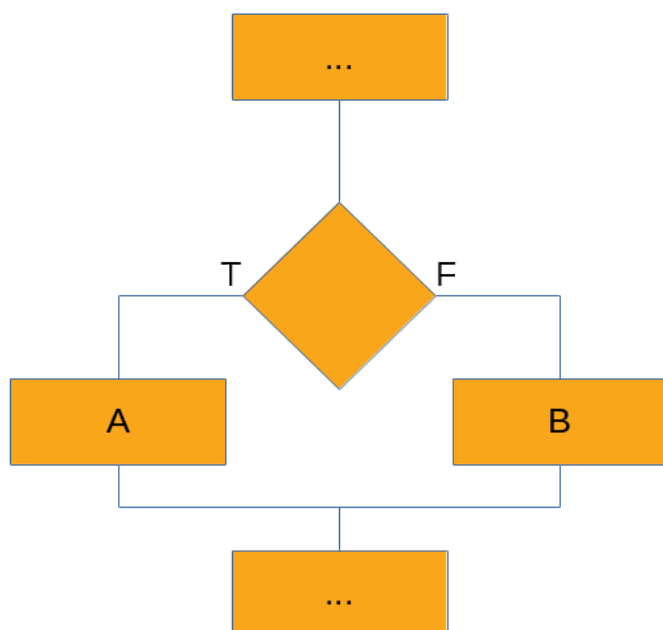
Od razu zaznaczamy, że jeśli te dodatkowe informacje mają się pojawić we właściwym przypadku (czyli przy spełnieniu warunku) na ekranie, to musimy pamiętać, żeby były wcięte na tym samym poziomie.

Aby uzyskać taki rezultat, program możemy napisać w sposób następujący:

```
1 print("Podaj liczbę aby uzyskać jej drugą potęgę:")
2 x = int(input())
3 if x % 3 == 0:
4     print("Liczba jest podzielna przez 3")
5     print("Wynik takiego dzielenia to", x//3)
6 if x % 4 == 0:
7     print("Liczba jest podzielna przez 4")
8     print("Wynik takiego dzielenia to", x//4)
9 if x == 13:
10    print("Podana liczba jest uważana za pechową")
11 print("Drugą potęgą tej liczby jest", x**2)
```

Powyższe ćwiczenie ma na celu lepsze objaśnienie uczniom roli wcięć. Warto zapytać uczniów, jak działałby program, gdyby linie 5 i 8 nie były wcięte (wtedy wykonywałyby się zawsze, niezależnie od podzielności liczb, gdyż nie są elementem bloku wykonywanego po spełnieniu warunku).

Następnie przechodzimy do omówienia instrukcji warunkowej if-else.



Odpowiadający kod w języku Python

```
...  
if warunek:  
    Blok instrukcji A  
else:  
    Blok instrukcji B  
...
```

Podstawową różnicą w stosunku do wcześniej prezentowanej wersji bez else, jest możliwość wykonania pewnych operacji, gdy warunek nie jest spełniony. Przykładowo możemy sprawdzić podzielność liczby przez 2 i poinformować użytkownika odpowiednio, że liczba jest parzysta lub nieparzysta. Nauczyciel prezentuje tę możliwość uczniom krótkim programem.

```
1 print("Podaj liczbę")  
2 x = int(input())  
3 if x % 2 == 0:  
4     print("Podałś liczbę parzystą")  
5 else:  
6     print("Podałś liczbę nieparzystą")
```

Po zaprezentowaniu przykładu prosimy uczniów o napisanie programu, który dla pewnej liczby x podanej przez użytkownika będzie wyliczał wartość następującego wyrażenia:

$$\frac{x+1}{x-3}$$

Program ma wykonywać obliczenia i wypisać wynik tylko, gdy  $x$  jest różne od 3 (czyli gdy mianownik nie jest zerem), w przeciwnym wypadku ma wypisać informację “Nie mogę wykonać dzielenia przez 0”. Przykładowe rozwiązanie ucznia mogłoby wyglądać następująco:

```
1 print("Podaj x")
2 x = int(input())
3 if x != 3:
4     result = (x + 1) / (x - 3)
5     print("Wynik:", result)
6 else:
7     print("Nie mogę wykonać dzielenia przez 0")
```

Po wykonaniu tego zadania przechodzimy do zaprezentowania bardziej rozbudowanej wersji instrukcji warunkowej – if-elif-else. Tym razem pojawia się nowe słowo kluczowe elif, które należy rozumieć jako “else if”. Wyobraźmy sobie sytuację, gdy mamy wiele możliwości (a nie tylko dwie jak w przypadku if-else), które musimy zidentyfikować sprawdzając kolejne warunki. Przykładowo sygnalizator świetlny dla pieszych może w danym momencie być w jednym z trzech stanów – świecić na czerwono, świecić na zielono albo migać na zielono (co informuje o zbliżającej się zmianie). Aby jednoznacznie ustalić, jaki był stan, z wykorzystaniem warunków, potrzebne jest przynajmniej sprawdzenie dwóch warunków. Przykładowo, na początku pytamy, czy światło było czerwone, jeśli nie, to pytamy, czy było migające zielone. Można też na początku zapytać, czy światło było migające zielone, a jeżeli nie, to zapytać, czy było czerwone. Schemat blokowy odpowiadający konstrukcji if-elif-else przedstawiono na następnej stronie.

Odpowiadający kod w języku Python

```
...
if warunek 1:
    Blok instrukcji A
elif warunek 2:
```

```
Blok instrukcji B  
else:  
    Blok instrukcji C  
...
```

Wyobraźmy sobie program w Pythonie sprawdzający, czy liczba jest dodatnia, ujemna lub nie jest żadną z nich (w przypadku zera). Program taki przedstawiamy uczniom.

```
1 print("Podaj x")  
2 x = int(input())  
3 if x > 0:  
4     print("Podałes liczbę dodatnią")  
5 elif x < 0:  
6     print("Podałes liczbę ujemną")  
7 else:  
8     print("Podana liczba nie jest liczbą ujemną ani dodatnią")
```

Następnie uczniowie otrzymują następujące zadanie:

Napisz program, który będzie zamieniał km/h na m/s lub m/s na km/h. Program ma na początku umożliwiać wybór typu konwersji, a potem pobierać wartość od użytkownika, którą będzie konwertował. Wynik programu ma być wyświetlany na ekranie.

Uczniowie zastanawiają się, jakie operacje muszą wykonać, żeby zamienić km/h na m/s. W razie problemów nauczyciel podpowiada rozwiązanie. Przy zamianie km/h na m/s liczbę należy pomnożyć przez 1000 (bo 1 km to 1000 m) i podzielić przez 3600 (bo 1h to 3600 s). Można to w prosty sposób rozpisać na tablicy:

$$x \frac{km}{h} = x \cdot \frac{1000 m}{60 min} = x \cdot \frac{1000 m}{60 \cdot 60 s} = x \cdot \frac{1000 m}{3600 s} = x \cdot \frac{1000}{3600} \cdot \frac{m}{s}$$

Przy zamianie m/s na km/h wykonujemy odwrotne operacje. Uczniom przekazujemy początek programu, który muszą uzupełnić:



```
1 print("Wybierz konwersję:")
2 print("1) km/h -> m/s")
3 print("2) m/s -> km/h")
4 option = input()
```

Następnie prosimy o uzupełnienie reszty. Zwracamy uwagę, że muszą rozważyć trzy możliwości – gdy wybrano pierwszą opcję, gdy wybrano drugą i gdy wybrana opcja jest niepoprawna (użytkownik wprowadza coś nieoczekiwanego). Przykładowe rozwiązanie problemu znajduje się poniżej.

```
1 print("Wybierz konwersję:")
2 print("1) km/h -> m/s")
3 print("2) m/s -> km/h")
4 option = input()
5
6 if (option == "1"):
7     print("Podaj prędkość w km/h:")
8     value = float(input())
9     value = value * 1000 / 3600
10    print("Prędkość w m/s to",value)
11 elif (option == "2"):
12    print("Podaj prędkość w m/s:")
13    value = float(input())
14    value = value * 3600 / 1000
15    print("Prędkość w km/h to",value)
16 else:
17    print("Niepoprawna opcja :( Kończę program")
```

Powyższy program da się usprawnić, jeśli chodzi o linie 9 i 14. Przykładowo, zamiast wykonywać dwie operacje – mnożenia przez 1000 i dzielenia przez 3600, możemy wykonać po prostu dzielenie przez 3.6, tak samo jak zamiast mnożenia przez 3600 i dzielenia przez 1000 możemy wykonać mnożenie przez 3.6.

### 3. Podsumowanie – 5 min.

Uczniowie prezentują nauczycielowi swoje rozwiązania. Nauczyciel wymienia wszystkie nowe elementy języka, które zostały poznane w ramach tej lekcji. Zwraca też uwagę na najczęściej pojawiające się błędy.