Mikrobitowcy

Autorzy: Filip Kłębczyk

Lekcja 2:

Pierwsze kroki z Pythonem. Instrukcja warunkowa

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

Cele lekcji:
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

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

Pojęcia kluczowe:
→ Python → instrukcja warunkowa → wcięcie → wcięty blok kodu → operator

Czas realizacji: 45 min.

Metody pracy:
- wykład,
- dyskusja,
- ćwiczenia praktyczne przy komputerze,
- prezentowanie efektów pracy,
- szukanie rozwiązań na postawiony problem.

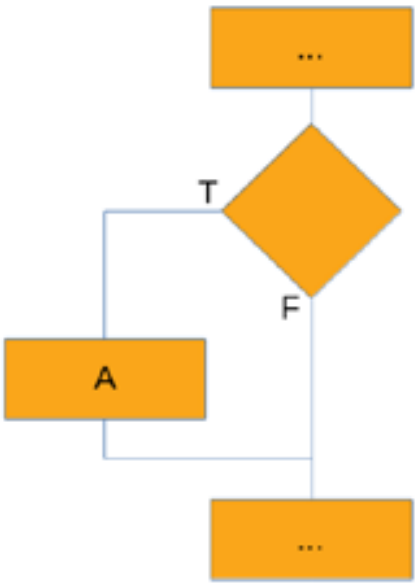
Treści programowe:
Podstawa programowa kształcenia ogólnego dla szkół podstawowych – II etap edukacyjny – klasy VII-VIII, informatyka:
I. Rozumienie, analizowanie i rozwiązywanie problemów. Uczeń:
3) przedstawia sposoby reprezentowania w komputerze wartości logicznych, liczb naturalnych (system binarny), znaków (kody ASCII) i tekstów; formułuje problem w postaci specyfikacji (czyli opisuje dane i wyniki) i wyróżnia kroki w algorytmicznym rozwiązywaniu problemów. Stosuje różne sposoby przedstawiania algorytmów, w tym w języku naturalnym, w postaci schematów blokowych, listy kroków;
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.
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;

Wprowadzenie w tematykę i integracja grupy (5 min.)

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

Część zasadnicza (35 min.)

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



Odpowiadający kod w języku Python

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

Tłumaczymy działanie: w momencie gdy warunek (wyrażenie logiczne) jest spełniony, 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 potęgę:")
2  x = int(input())
3  if x % 3 == 0:
4      print("Liczba jest podzielna przez 3")
5  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 sprawdzenie reszty z dzielenia przez 3, a następnie z wykorzystaniem operatora == sprawdzenie, czy jest ona równa 0. Jeśli warunek jest spełniony, 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 wnętrzu funkcji/klasy, jest oznaczany nawiasami klamrowymi lub słowami kluczowymi (np. begin-end). Natomiast w Pythonie 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 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
```

Natomiast 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, by poza podzielnością przez 3 sprawdzał dodatkowo też"

- podzielność przez 4 i informował o tym
- 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 liczbe aby uzyskać jej druga potege:")
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("Druga potęga tej liczby jest", x**2)
```

Następnym zadaniem, jakie proponujemy uczniom, jest rozbudowanie programu, tak 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:

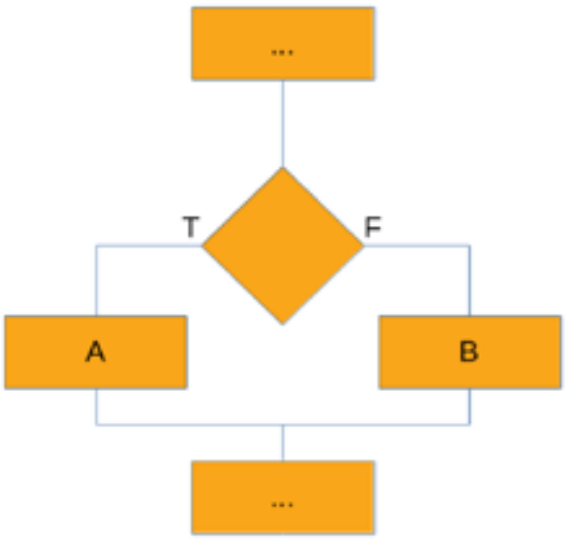
```
Liczba jest podzielna przez 3
Wynik takiego dzielenia to 8
Liczba jest podzielna przez 4
Wynik takiego dzielenia to 6
Drugą potęgą tej liczby jest 576
```

Od razu zaznaczamy, że jeśli te dodatkowe informacje mają się pojawić we właściwym przypadku (czyli przy spełnieniu warunku), musimy pamiętać o właściwym zastosowaniu wcięć:

```
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 uświadomienie 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.

Następnym zadaniem, jakie proponujemy uczniom, jest rozbudowanie programu, tak 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:

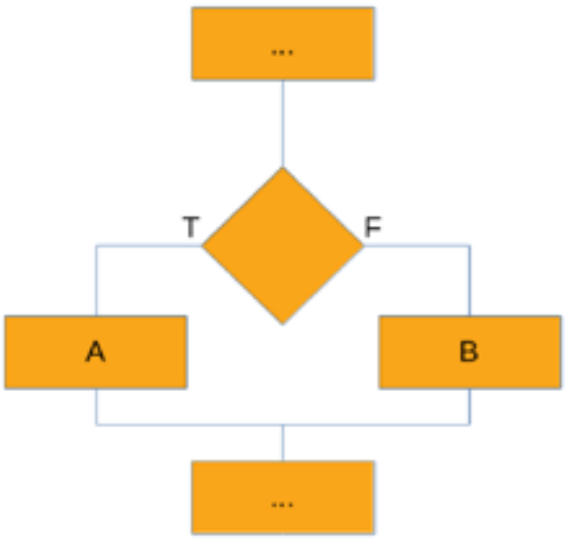
```
Liczba jest podzielna przez 3
Wynik takiego dzielenia to 8
Liczba jest podzielna przez 4
Wynik takiego dzielenia to 6
Drugą potęgą tej liczby jest 576
```

Od razu zaznaczamy, że jeśli te dodatkowe informacje mają się pojawić we właściwym przypadku (czyli przy spełnieniu warunku), musimy pamiętać o właściwym zastosowaniu wcięć:

```
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 uświadomienie 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("Podałeś liczbę parzystą")
5  else:
6      print("Podałeś liczbę nieparzystą")
```

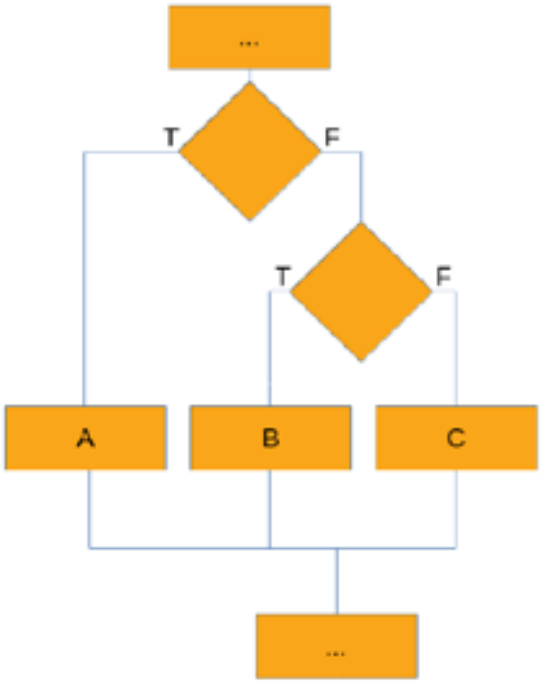
Po zaprezentowaniu przykładu prosimy uczniów o napisanie programu, który dla pewnej liczby x podanej przez użytkownika będzie wyliczał wartość następującego wyrażenia:

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

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 - mieć światło czerwone, mieć światło zielone albo migające zielone (informujące 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, pytamy czy było migające. Można też na początku zapytać, czy było migające i jeżeli nie, to zapytać czy było czerwone. Schemat blokowy odpowiadający konstrukcji if-elif-else:



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.

```
print("Podaj x")
x = int(input())
if x > 0:
    print("Podałeś liczbę dodatnią")
elif x < 0:
    print("Podałeś liczbę ujemną")
else:
    print("Podana liczba nie jest liczbą ujemną ani dodatnią")
```

Następnie, po zaprezentowaniu powyższego kodu, 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 ma wyświetlać na ekranie. Uczniowie zastanawiają się, jakie operacje muszą wykonać, żeby zamienić km/h na m/s. razie problemów nauczyciel podpowiada rozwiązanie. Przy zamianie km/h na m/s liczbę należy pomnożyć przez 1000 (bo 1 km to 1000 m) i 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:

```
print("Wybierz konwersję:")
print("1) km/h ->  m/s")
print("2) m/s  -> km/h")
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.

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

if (option == "1"):
    print("Podaj prędkość w km/h:")
    value = float(input())
    value = value * 1000 / 3600
    print("Prędkość w m/s to",value)
elif (option == "2"):
    print("Podaj prędkość w m/s:")
    value = float(input())
    value = value * 3600 / 1000
    print("Prędkość w km/h to",value)
else:
    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.

Podsumowanie (5 min.)

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