Wybrane elementy informatyki - wykład 5

Podstawy programowania w języku PYTHON

dr Marcin Ziółkowski

Instytut Matematyki i Informatyki
Akademia im. Jana Długosza w Częstochowie

6 kwietnia 2017 r.

JĘZYK PYTHON

Język Python jest jednym z najmłodszych, ale zarazem najczęściej używanych obecnie języków programowania. Ma dosyć łatwą składnię, stosunkowo niewiele słów kluczowych, a także bardzo bogatą bazę bibliotek, z pomocą których można tworzyć nawet bardzo skomplikowane projekty programistyczne. O jego popularności i możliwościach może świadczyć fakt, że języka tego używają programiści na całym świecie. Jego popularność stale rośnie i język ten obok języka C++ i języka JAVA staje się jednym z najczęściej wybieranych języków programowania zarówno przy tworzeniu aplikacji na komputery osobiste jak i na urządzenia mobilne. Język ten jest wieloplatformowy (wykorzystywany w różnych systemach operacyjnych), a jego podstawowe wersje są elementem składowym darmowych systemów operacyjnych takich jak: LINUX, SOLARIS czy BSD. Język Python jest uważany również za wspaniały język do początkowej nauki programowania z uwagi na jego prostotę i logiczną składnię. Używanie, w początkowej fazie nauki programowania, właśnie tego języka uczy dobrych nawyków zapisu programów komputerowych.

JĘZYK PYTHON

Język Python ma bogate możliwości zarówno programowania proceduralnego jak i obiektowego. Jego zaletą jest również to, że słowa kluczowe używane w tym języku są identyczne jak w innych nowoczesnych językach wysokopoziomowych takich jak: C++, JAVA czy PHP. Jednak w porównaniu z tymi językami tworzenie programów jest bardziej intuicyjne i nie wymaga od początkującego użytkownika szerokiej wiedzy informatycznej i pamiętania wielu edycyjnych szczegółów. Z drugiej strony, w języku tym występują ciekawe rozwiązania, których inne języki nie posiadają m.in. istnienie obliczeniowego typu liczb zespolonych, operatora potęgowania, domyślnego typu wprowadzanych danych jako danych typu string czy bardzo wygodnej struktury danych: list, a także w zasadzie nieograniczonego zakresu danych liczbowych.

Sprawia to, że język ten można również wykorzystać jako wspaniały kalkulator. Język Python posiada również bogate biblioteki graficzne, co daje możliwości tworzenia interfejsów graficznych. Jednym z minusów tego języka jest to, że jest to język interpretowany a nie kompilowany, co powoduje wolniejsze działanie programów, lecz z drugiej strony pozwala na szybsze znajdowanie błędów na etapie tworzenia programu.

PROGRAM W JĘZYKU PYTHON

Program napisany w języku python jest ciągiem instrukcji. Oznacza to, że ważne są nie tylko instrukcje zawarte w programie, ale także ich kolejność. Instrukcje zawsze zaczynają się małą literą. Program w języku Python piszemy w pliku tekstowym - możemy więc do tego celu użyć dowolnego edytora tekstowego (np. NOTATNIKA). Musimy go jednak zapisać z rozszerzeniem py. Należy dodatkowo pamiętać, że sam plik tekstowy nie jest jeszcze programem!! Plik tekstowy programu należy wykonać. W przypadku języków interpretowanych plik tekstowy jest tłumaczony linijka po linijce i w trakcie tłumaczenia wykonywany przy użyciu specjalnego programu zwanego interpreterem. Najwygodniej jest jednak zainstalować środowisko programistyczne, które zawiera w sobie nie tylko edytor do pisania programów, ale również interpreter, zbiór potrzebnych bibliotek oraz interfejs graficzny, za pomocą którego w łatwy sposób można kompilować, poprawiać i uruchamiać programy. Dla systemu WINDOWS najlepiej pobrać instalator środowiska ze strony **www.python.org**. Obecna wersja języka to wersja 3.5.

SZKIELET PROGRAMU W JĘZYKU PYTHON

Spróbujemy teraz zapoznać się z podstawowymi zasadami pisania programów w języku PYTHON. Dla przykładu napiszemy pierwszy program, który wypisuje na ekranie komputera komunikat : "I LOVE PROGRAMMING IN PYTHON". Oto plik źródłowy takiego programu:

PROGRAM 1 - WYPISYWANIE TEKSTU NA EKRANIE

```
print("I LOVE PROGRAMMING IN PYTHON")
input()
```

- Pierwszą instrukcją programu jest instrukcja print("I LOVE PROGRAMMING IN PYTHON") odpowiadająca za wypisanie tekstu na ekranie oraz automatyczne (domyślne) przejście do nowej linii.
- Linijka input() pozwala na zatrzymanie działania programu do momentu wciśnięcia przez użytkownika dowolnego klawisza. Ta linia jest obowiązkowa w systemie operacyjnym WINDOWS. W innych systemach operacyjnych można tą linię pominąć.

Rozważmy teraz przykładowy program, który dodaje dwie liczby całkowite i wyświetla wynik na ekranie. Oto kod programu:

PROGRAM 2 - DODAWANIE LICZB CAŁKOWITYCH

```
print("Podaj pierwsza liczbe")
a=int(input())
print("Podaj druga liczbe")
b=int(input())
print("Suma liczby",a,"oraz",b,"wynosi",a+b)
input()
```

OPERATORY ARYTMETYCZNE I LOGICZNE ORAZ TYPY DANYCH

W języku PYTHON mamy do czynienia najczęściej z następującymi typami danych:

1. int - liczby typu całkowitego
2. float - liczby zmiennoprzecinkowe (rzeczywiste)
3. complex - liczby zespolone
4. string - łańcuchy znaków (np. słowa) - jest to domyślny typ wprowadzanych danych

OPERATORY ARYTMETYCZNE I LOGICZNE ORAZ TYPY DANYCH

W języku python używamy następujących operatorów arytmetycznych : $+, -, *, /, \%$ (reszta z dzielenia), $**$ (potęgowanie) oraz znaków relacyjnych: $=, ==, <, >, <=, >=$. Należy odróżnić operację podstawienia np. $a = 2$ od operacji porównania np. $a == 2$ (czy a=2?). Podwójna równość jest używana w instrukcjach warunkowych i pętlach.
W języku Python używamy następujących operatorów logicznych : ! (negacja - zaprzeczenie), *or* (alternatywa - "lub") oraz *and* (koniunkcja - "i"). Natomiast zapis $! =$ oznacza "jest różne".

INSTRUKCJA WARUNKOWA IF

Rozważmy teraz przykładowy program, który dzieli dwie liczby rzeczywiste i wyświetla wynik na ekranie. Oto kod programu:

PROGRAM 3 - DZIELENIE LICZB RZECZYWISTYCH

```
print("Podaj pierwsza liczbe")
a=float(input())
print("Podaj druga liczbe")
b=float(input())
if b==0:
   print("Ale z Ciebie matematyk! Nie dzielimy przez zero!")
else:
    print(a,":",b,"=",a/b)
input()
```

Zauważmy, że po instrukcji if oraz instrukcji else występuje dwukropek i obowiązkowe wcięcie na ustaloną liczbę spacji (zwykle 4 spacje).

WCZYTYWANIE ŁAŃCUCHÓW ZNAKÓW - TYP STRING

PROGRAM 4 - WCZYTYWANIE ŁAŃCUCHÓW ZNAKÓW - TYP STRING

```
print("Podaj login")
a=input()
print("Podaj haslo")
b=input()
if a=="Marcin" and  b=="AJD":
    print("Zostales zalogowany")
else:
    print("Bledny login lub haslo")
input()
```

INSTRUKCJA WARUNKOWA IF, ELIF

PROGRAM 5 - OBLICZENIA Z KWADRATEM

```
print("Podaj dlugosc boku kwadratu:")
a=float(input())
if a<=0:
    print("Kwadrat nie istnieje")
else:
    print("Co chcesz obliczyc: 1. POLE  2. OBWOD")
    wybor=int(input())
    if wybor==1:
        print("Pole wynosi: ",a*a)
    elif wybor==2:
        print("Obwod wynosi: ",4*a)
    else:
        print("Zly wybor!")
input()
```

FUNKCJE MATEMATYCZNE

Język PYTHON zawiera definicje najważniejszych stałych i funkcji matematycznych. Aby ich używać trzeba na początku programu zadeklarować użycie modułu math. Oto najważniejsze funkcje:

- math.pi - wartość liczby π
- math.sqrt(x) - pierwiastek kwadratowy
- math.abs(x) - wartość bezwzględna
- math.log(x) - logarytm naturalny
- math.sin(x), math.cos(x), math.tan(x) - funkcje trygonometryczne

MODUŁ MATH

PROGRAM 6 - FUNKCJE TRYGONOMETRYCZNE

```
import math
print("Podaj miare kata w stopniach:",end="")
a=float(input())
x=(math.pi*a)/180
print("sin(",a,") wynosi",math.sin(x))
print("cos(",a,") wynosi",math.cos(x))
print("tg(",a,") wynosi",math.tan(x))
input()
```

INSTRUKCJE ITERACYJNE

Instrukcje iteracyjne w każdym języku programowania służą do realizacji algorytmów z powtórzeniami. W algorytmach takich występują wielokrotne realizacje tych samych instrukcji. Instrukcje iteracyjne nazywane są również często pętlami. Każdy język programowania ma własny zestaw instrukcji iteracyjnych. W języku PYTHON mamy do czynienia z następującymi instrukcjami iteracyjnymi:

1. pętla WHILE
2. pętla FOR

Zapoznamy się teraz ze składnią każdej z pętli oraz pokażemy na wybranych przykładach ich zastosowanie.

PĘTLA WHILE

Instrukcja iteracyjna WHILE jest podstawową i najważniejszą instrukcją iteracyjną w języku Python. W zasadzie wszystkie programy można tworzyć używając jedynie tej instrukcji. Jednak z pewnych względów praktycznych używamy również pętli FOR, która jest modyfikacją składniową pętli WHILE.
Schemat instrukcji while jest następujący:

SCHEMAT PĘTLI WHILE

```
while WARUNEK KONTYNUACJI PĘTLI:
    instrukcja 1
    instrukcja 2
    ..............
```

Zauważmy, że podobnie jak przy instrukcji warunkowej IF-ELIF konieczny jest dwukropek i obowiązkowe wcięcie na ustaloną liczbę spacji.

BŁĘDY ZWIĄZANE Z INSTRUKCJĄ WHILE

Najczęściej popełnianym przez początkujących programistów błędem jest złe ustalenie warunku kontynuacji pętli. Często pętla nigdy nie zadziała (warunek kontynuacji pętli nigdy nie jest spełniony) lub pętla nigdy się nie zakończy (warunek kontynuacji pętli jest spełniony zawsze). Takie błędy powodują nie tylko złe działanie programów, ale również (w przypadku nieskończonych pętli) niepotrzebne obciążanie zasobów komputera (pamięć RAM) i w efekcie niestabilne działanie systemu operacyjnego.
Nieskończone pętle iteracyjne są natomiast wykorzystywane przy tworzeniu wirusów komputerowych. W normalnych sytuacjach trzeba unikać takich błędów.
Zauważmy również, że ilość powtórzeń pętli może być dana z góry (np. pięciokrotne wyświetlenie ostrzeżenia) lub zależeć od spełnienia pewnego warunku (np. wczytujemy liczby z klawiatury dopóki użytkownik nie wprowadzi liczby ujemnej).
Zaprezentujemy teraz kilka przykładów wykorzystania pętli while.

PRZYKŁADY ZASTOSOWANIA PĘTLI WHILE

Spróbujmy napisać program, który pięciokrotnie wyświetla na ekranie napis: ”I love PYTHON”.

PROGRAM 7 - WIELOKROTNE WYPISYWANIE TEKSTU NA EKRANIE

```
import os              # importowanie biblioteki os
os.system("cls") # czyszczenie ekranu
i=1                 # wprowadzamy "licznik" wyświetleń napisu
while i<=5:        # pętla kończy się, gdy i>5
    print("I love PYTHON")
    i=i+1          # zwiększamy licznik
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI WHILE

Teraz napiszemy program, który oblicza wartość sumy $1 + 2 + 3 + \ldots + 10000$.

PROGRAM 8 - OBLICZANIE SUMY

```
i=1
suma=0                  #wprowadzamy zmienną suma
while i<=10000:
    suma=suma+i         #obliczamy aktualną sumę
    i=i+1
print("Suma wynosi",suma) #wyświetlamy sumę
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI WHILE

Jest oczywiste, że w pętli WHILE może znajdować się instrukcja warunkowa IF-ELIF, a także inna pętla WHILE. Pokażemy to na kolejnych przykładach.

PROGRAM 9 - OBLICZANIE ILOCZYNU LICZB PODZIELNYCH PRZEZ 3 Z PRZEDZIAŁU [1,20]

```
iloczyn=1 # zmienna przechowuje iloczyn
i=3
while i<=20:
    if i%3==0:
        iloczyn=iloczyn*i #obliczamy iloczyn
    i=i+1
print("Iloczyn wynosi",iloczyn)
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI WHILE

Napiszemy program, który dzieli dwie liczby rzeczywiste, wymuszając poprawne dane wejściowe.

PROGRAM 10 - DZIELENIE LICZB RZECZYWISTYCH

```
a=float(input("Podaj I liczbe:")) #wprowadzanie szybciej
b=float(input("Podaj II liczbe:"))
while b==0:
    print("Dzielnik jest równy zero. Podaj go jeszcze raz")
    b=float(input("Podaj II liczbe:"))
print("Iloraz wynosi",a/b)
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI WHILE

A teraz program logujący użytkownika.

PROGRAM 11 - LOGOWANIE

```
import os
login=input("Podaj login:")
haslo=input("Podaj haslo:")
while login!="marcin" or haslo!="AJD":
    os.system("cls")
    print("Nie zgadza się! Wprowadź jeszcze raz")
    login=input("Podaj login:")
    haslo=input("Podaj haslo:")
print("Witaj MARCIN. Zostales zalogowany!")
input()
```

PĘTLA FOR

Instrukcja iteracyjna FOR jest pewną składniową modyfikacją pętli WHILE. Jest bardzo lubiana przez programistów z uwagi na krótszy zapis. Jest szczególnie prosta w przypadku pętli o znanej liczbie powtórzeń. W języku Python jej konstrukcja jest połączona z matematyczną ideą ciągu arytmetycznego.

SCHEMAT PĘTLI FOR

```
for zmienna in range(POCZ. WAR. ZM.,BRZEGOWA WAR. ZM.,KROK):
    instrukcja 1
    instrukcja 2
    ............
```

UWAGA. Nie wszystkie składowe funkcji RANGE muszą wystąpić.

PRZYKŁADY ZASTOSOWANIA PĘTLI FOR

Spróbujemy wytłumaczyć teraz zastosowanie funkcji range. Poniższy program wypisze liczby całkowite mniejsze od 6, czyli: 0,1,2,3,4,5.

CIĄG 1

```
for i in range(6):
    print(i)
input()
```

Ten program wypisze liczby całkowite mniejsze od 10 i większe lub równe 2, czyli: 2,3,4,5,6,7,8,9.

CIĄG 2

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

PRZYKŁADY ZASTOSOWANIA PĘTLI FOR

Ten program wypisze liczby całkowite mniejsze od 20 i większe lub równe 2, ale w odstępie co trzecią, czyli: 2,5,8,11,14,17.

CIĄG 3

```
for i in range(2,20,3):
    print(i)
input()
```

Ten program wypisze liczby całkowite mniejsze równe od 20 i większe od 3, ale w odstępie co drugą (do tyłu), czyli: 20,18,16,14,12,10,8,6,4.

CIĄG 4

```
for i in range(20,3,-2):
    print(i)
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI FOR

Spróbujmy napisać program, który wielokrotnie wyświetla na ekranie napis: "I love PYTHON" z wykorzystaniem pętli for.

PROGRAM 12 - WIELOKROTNE WYPISYWANIE TEKSTU NA EKRANIE - PĘTLA FOR

```
n=int(input("Ile razy piszemy?"))
for i in range(n):
    print("I love PYTHON")
input()
```

PRZYKŁADY ZASTOSOWANIA PĘTLI FOR

Spróbujmy napisać program, który rysuje wypełniony kwadrat o wymiarach $n \times n$.

PROGRAM 13 - KWADRAT

```
n=int(input("Podaj rozmiar kwadratu:"))
for i in range(n):
    for j in range (n):
        print("*   ",end="")
    print("\n")
input()
```

KILKA INFORMACJI O ŁAŃCUCHACH ZNAKÓW

Jak to było przedstawione na poprzednim wykładzie, domyślnym typem zmiennych dla PYTHONA jest zmienna typu string. Z drugiej strony napisy (łańcuchy znaków) mogą być rozumiane jako wektory składające się z pojedynczych znaków. Pozwala to na pisanie programów modyfikujących napisy, obliczających liczbę liter itd. Oto przykład programu, który wypisuje wprowadzony napis od końca:

PROGRAM 14 - NAPIS OD KOŃCA

```
s=input("Podaj napis:")
for i in range(len(s)-1,-1,-1):
    print(s[i],end="")
input()
```