pyladies

Siadamy co drugi rząd!

pyladies

4.7

Wielkiej powtórki ciąg dalszy

- zadania -

pyladies

Directed by: Mirosław Horbaczewski

10.01.2018

pyladies

WIFI:
PUT-events-WiFi
Login: user_69211
Hasło: 9my3ci6kaZI

CO NAS DZISIAJ CZEKA

Powtórzymy materiał z poprzednich zajęć, wykorzystując zdobytą dotychczas wiedzę do rozwiązywania mniej lub bardziej skomplikowanych zadań w interpreterze Python:

1. Operatory logiczne
2. Instrukcje warunkowe
3. Listy
4. Pętle (for, while)

Warunki logiczne – inne spojrzenie

Przypomnienie z 1 klasy liceum:

Koniunkcja

p	q	p and q
1	1	1
1	0	0
0	1	0
0	0	0

Wniosek:

warunek jest prawdziwy tylko kiedy oba zdania są prawdziwe

Warunki logiczne – c.d.

Koniunkcja
3 wyrażeń (zdań)

p	q	r	p and q and r
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Wniosek:

warunek jest prawdziwy gdy wszystkie zdania są prawdziwe

Warunki logiczne – c.d.

Alternatywa

p	q	p or q
1	1	1
1	0	1
0	1	1
0	0	0

Wniosek:

warunek jest prawdziwy gdy przynajmniej jedno zdania jest prawdziwe

Warunki logiczne – c.d.

Alternatywa

3 warunków (zdań)

p	q	r	p or q or r
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	1
0	1	1	1
0	1	0	1
0	0	1	1
0	0	0	0

Wniosek:

warunek jest prawdziwy gdy przynajmniej jedno zdanie jest prawdziwe

A jak to jest w Pythonie?

Przykłady reprezentacji wartości logicznych w programie:

1. Zdania prawdziwe (wartość logiczna w tabelce to 1)

- **łańcuchy (stringi):** `'Ala ma kota'` `"Ala ma kota"`
- **liczby (integer, float, zespolone):** `1` `-25` `3.14159` `2+7j`
- **listy:** `[1,2,3]` `['Ala ma kota', 'Ala', 'ma', 'kota']`

2. Zdania fałszywe (wartość logiczna w tabelce to 0)

- **łańcuchy (stringi):** `''`
- **liczby (integer, float, liczby zespolone):** `0` `0.0` `0j`

Wartości logiczne – w praktyce Pythona

```
>>> bool("Ala ma kota")
True
>>> bool("")
False
>>> bool(" ")
True
>>> bool(3.14)
True
>>> bool(1)
True
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool(['element listy'])
True
>>> bool([])
False
>>>
>>>
```

Przypomnienie z poprzednich zajęć – wyrażenia złożone

W wyrażeniach logicznych operatory **or** oraz **and** działają od strony lewej do prawej.

```
x = 7
if  ( x > 10 or x <= 5 and x > 15 ):
    print("Prawda")
else:
    print("Fałsz")
```

Pomocne może być stosowanie nawiasów dla uniknięcia błędów.

Przykład

Może być zwrócona wartość logiczna (**True**, **False**) lub składnik złożonego wyrażenia (lista, liczba, łańcuch znaków).

```
x = 15
x > 10  and  ""
```

- zwraca "" , bo powyższy warunek jest fałszywy ale mamy wartość po prawej

Ale:

```
x = 15
x > 10  and  x > 20
```

- zwraca **False** , bo powyższy warunek jest fałszywy ale mamy również warunek po prawej stronie operatora logicznego

Ciekawostka

x and y

jeżeli **x** jest fałszem, wtedy wyrażenie przyjmuje wartość **x**, w przeciwnym wypadku wartość **y**

```
[ ]  and  35
[ 'prawda' ]  and  35
```

x or y

jeżeli **x** jest fałszem, wtedy wyrażenie przyjmuje wartość **y**, w przeciwnym wypadku **x**

```
[ ]  or  35
[ 'prawda' ]  or  35
```

Przypomnienie sprzed tygodnia (and)

W przypadku operatora **and**:

jeżeli wartość po jego lewej stronie zostanie zinterpretowana jako True:

ZAWSZE zostanie zwrócona wartość po prawej stronie.

Inaczej:

1. Jeśli wszystkie wyrażenia są prawdą w kontekście logicznym, zwróć wartość ostatniego wyrażenia.

2. Jeśli któreś z wyrażeń jest fałszem, zwróć wartość pierwszego fałszywego wyrażenia.

Zaczynamy od lewej strony!

Przypomnienie sprzed tygodnia (or)

Operator **or** działa od lewej do prawej podobnie jak **and**!

1. Jeśli któreś wyrażenie jest prawdą **or** zwraca jego wartość natychmiast.
2. Jeśli wszystkie wyrażenia są fałszem **or** zwraca wartość ostatniego wyrażenia.

Przykłady

```
'' and True and [1, 2, 3]                    #       ''

'kot' and True or [1, 2, 3]                  #       True

'' or True or []                             #       True

'' or True and []                            #       []

[1] or [] and ''                             #       [1]

True and False and ()                        #       False

 'PyTraining' or 'Disco' and 'Beer' or 'Wine'       #       'PyTraining'
```

Sprintem przez pętle

for

```
for element in sekwencja:

    #  jakiś kod
```

while

```
while warunek:

    # jakiś kod (drukowanie, liczenie, itp.)
```

Zadanie 1

Program prosi użytkownika o wpisanie w konsoli dowolnego zdania.

Naszym zadaniem jest wyświetlenie wszystkich samogłosek , występujących w tym zdaniu.

Dla bardziej ambitnych ;-)

- dodatkowo: ile razy każda samogłoska występuje we wpisanym zdaniu

Zadanie 1 – wskazówka 1

1. Musimy zdefiniować zmienną, w której przechowywane będą wszystkie samogłoski występujące w alfabecie łacińskim czy angielskim (dla uproszczenie odrzucamy te z polskimi „ogonkami”)

2. Do tego celu doskonale nadaje się typ danych: lista
   Wystarczy odwołać się tylko do jednej zmiennej, żeby mieć dostęp do różnych interesujących nas wartości

3. Elementy listy są rodzajem stałej, bo liczba samogłosek w danym alfabecie nie zmienia się. Nie jest to obowiązkowe, ale zwyczajowo przyjmuje się, że stałe zapisujemy dużymi literami. Ponadto ze względu na możliwość łatwiejszego uzyskania pomocy na forach internetowych nie tylko polskich, warto zapisywać nazwy zmiennych w języku angielskim.

Zadanie 1 – początek rozwiązania

```
VOWELS = ['a','e','i','o','u','y']

word = input('Wpisz dowolne zdanie: ')
```

Uwaga:

Lista to typ sekwencyjny

(czyli elementy listy są uporządkowane, można się do nich odwołać wg pozycji występowania danego elementu, inaczej mówiąc według indeksu, ważne - numerowanego od zera, nie od jedynki; podobieństwo do tablicy w innych językach programowania)

np. VOWELS[0] = ‘a’

VOWELS[5] = ‘y’

Zadanie 1 – wskazówka 2 - pętla for

Do rozwiązania problemu idealna będzie pętla for, bo przebiega przez całe zdanie i wypisuje z niego tylko interesujące nas samogłoski.

Ale czy tak?

```
for letter in word:
    print(letter)
```

Widać, że nie o to nam chodziło

Wpisz dowolne zdanie: Ala ma kota

A
l
a

m
a

k
o
t
A

Zadanie 1 – kolejna wskazówka

Zatem dodatkowo w kodzie musi pojawić się instrukcja, która odfiltruje zbędne litery, zostawiając tylko samogłoski.

Nic prostszego, wypisywana litera musi być samogłoską, czyli musi należeć do listy samogłosek VOWELS.

Użyjemy instrukcji warunkowej:

if letter **in VOWELS**:

Pamiętamy o dwukropku na końcu warunku oraz odpowiednich wcięciach

Zadanie 1 – ostatnia wskazówka

Kod programu przyjmie więc postać:

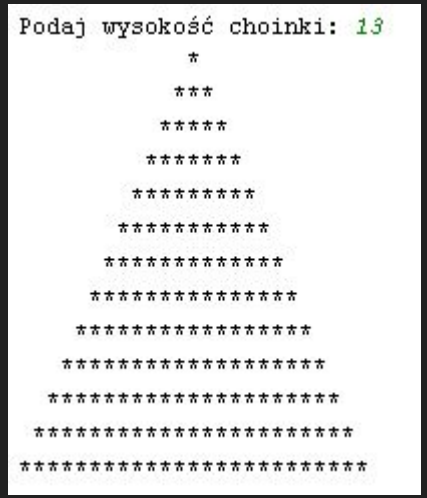
```
VOWELS = ['a','e','i','o','u','y']
word = input('Wpisz dowolne zdanie: ')
for letter in word:
    if letter in VOWELS:
        print(letter)
```

Zadanie 1

Pytania?

Zadanie 2 - choinka

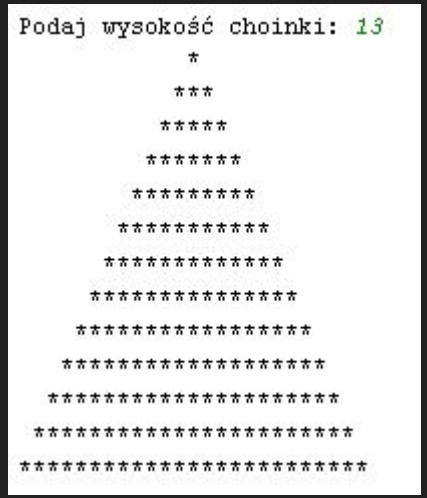
W związku z niedawno zakończonymi świętami rysujemy choinkę złożoną ze spacji i gwiazdek.
Pamiętamy, że należy najpierw zapytać o wysokość choinki.
Przykład:



Zadanie 2 - podpowiedź

Na co zwrócić uwagę planując algorytm rozwiązania?

- wolnym wierszu nie ma spacji
- w każdym następnym wierszu idąc w górę dochodzi jedna spacja na początku (końcowe są oczywiste, więc nas nie interesują)
- w każdym następnym wierszu idąc w górę są dwie gwiazdki mniej



Zadanie 2 – przykładowe rozwiązanie

```
rozmiar_choinki = int(input("Podaj wysokość choinki: "))

liczba_spacji_na_poczatku = rozmiar_choinki - 1
liczba_gwiazdek = 1

while liczba_spacji_na_poczatku >= 0:
    print(' ' * liczba_spacji_na_poczatku + '*' * liczba_gwiazdek)
    liczba_spacji_na_poczatku -= 1
    liczba_gwiazdek += 2
```

Zadanie 3

Wyliczyć sumę ciągu wszystkich liczb parzystych
wybranych
spośród pierwszych **N** wyrazów ciągu Fibonacciego.

Definicja ciągu Fibonacciego

Ciąg zaczyna się od liczb: 0 i 1.

Każdy inny dowolny wyraz ciągu jest sumą dwóch wyrazów go poprzedzających.

Przykład:

Pierwsze kilkanaście wyrazów ciągu:

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, ...

Zatem parzyste składniki ciągu:

0, 2, 8, 34, 144, 610, ...

Zadanie 3 – wskazówka 1

Dla przypomnienia wzór ze szkoły średniej na dowolny wyraz ciągu Fibonacciego:

$a_0 = 0$

$a_1 = 1$

.....

$a_n = a_{n-2} + a_{n-1}$

Zadanie 3 – wskazówka 2 – wariant normalny (bez parzystości)

```
# wprowadzamy dane początkowe:

N = int(input("Podaj liczbę wyrazów ciągu Fibonacciego: "))

# musimy pamiętać o operatorze rzutowania int (zamiana stringu na liczbę)

fib = [0,1]     # dwuelementowa lista początkowa
i = 2       # bo zaczynamy obliczenia od drugiego wyrazu ciągu
            # (zerowy i pierwszy podaje nam definicja)
```

Zadanie 3 – wskazówka 3

```
# budowanie listy zawierające wszystkie wyrazy ciągu Fibonacciego o zadanej długości:


while i < N:

    nowy = fib[i-2] + fib[i-1]    # stosujemy wzór z definicji dla każdego następnego składnika

    fib.append(nowy)     # powiększamy listę wyrazów ciągu o kolejne elementy
                         # wyliczone z powyższego wzoru

    i += 1

# ważne, żeby licznik zwiększać na końcu pętli (wtedy nie przekraczamy liczby N wyrazów ciągu)
```

Zadanie 3 – rozwiązanie bez parzystości

```
# wprowadzamy dane początkowe:
N = int(input("Podaj liczbę wyrazów ciągu Fibonacciego: "))
# musimy pamiętać o operatorze rzutowania int (zamiana stringu na liczbę)
fib = [0,1]
i = 2
#print(fib[0])
#print(fib[1])
#print(fib)
# budowanie listy zawierające wszystkie wyrazy ciągu Fibonacciego o zadanej długości:
while i < N:
    nowy = fib[i-2] + fib[i-1]
    fib.append(nowy)
    i += 1
print(fib)
```

Zadanie 3 - wskazówki

1. Uzyskanie parzystości:
   - dzielenie modulo 2, czyli
   reszta z dzielenia przez 2: 0 dla liczb parzystych,
   1 dla nieparzystych
   operator %
2. Sprawdzenie warunku instrukcją if
3. Sumowanie elementów listy: sum(lista)

Rozwiązanie

```
N = int(input("\nPodaj liczbę wyrazów ciągu Fibonacciego: "))
fib = [0,1]
parzyste = [0]
i = 2

while i < N:
    nowy = fib[i-2] + fib[i-1]
    fib.append(nowy)
    if nowy % 2 == 0:
        parzyste.append(nowy)
    i += 1
suma = sum(parzyste)

print("\nWszystkie wyrazy ciągu Fibonacciego:")
print(fib)
print("\nTylko parzyste:")
print(parzyste)
print("\nSuma:", suma)
```

ZADANIE DOMOWE

Postaram się coś wrzucić w niedzielę na stronę:

http://PyLadies.info


QUARTERS ONLY
25¢
1. DROP COIN IN SLOT.
2. PULL HANDLE DOWN ALL THE WAY-LET GO
3. TO CLEAR VISION - TURN RED KNOB.
2414

Q&A

Feedback

[link do aktualnego feedback form będzie podany na FB]

Stay in touch

- materiały i zadania: pojawią się pod wydarzeniem po zajęciach
- grupa na FB: goo.gl/GLiX1V
- fanpage: facebook.com/pyladiespoznan/
- dodatkowo ta prezentacja na stronie: PyLadies.info (ale pewnie dopiero w weekend)