rozdział 4:
ZMIENNE I INSTRUKCJE

ostatnia modyfikacja: 04.12.18

Podstawy programowania

definicja:

- **zmienna** to nazwany kontener służący do przechowywania danych
- na razie skupimy się na zmiennych przechowujących dane liczbowe i logiczne

Podstawy programowania

nazwa zmiennej:

- może zawierać litery (małe i wielkie), cyfry i znak _ (podkreślenie)
- nie może zaczynać się od cyfry
- może zawierać znaki narodowe
- litery małe i wielkie traktowane są jako różne

Podstawy programowania

- niektóre słowa **nie mogą** być użyte jako nazwy zmiennych, ponieważ są **zastrzeżone** do wyłącznego użytku przez Pythona
- są to tzw. **słowa kluczowe** (ang. *keywords*)

Podstawy programowania

- oto **słowa kluczowe** Pythona 3:

```
False      class      finally    is         return
None       continue   for        lambda     try
True       def        from       nonlocal   while
and        del        global     not        with
as         elif       if         or         yield
assert     else       import     pass
break      except     in         raise
```

Podstawy programowania

nazwa zmiennej:

- nazwy zmiennych powinny być **samokomentujące** – tzn. z ich nazwy powinno wynikać, jaką daną przechowują
- zmienne **mogą** występować w wyrażeniach na takich samych zasadach jak literały
- jednak użycie zmiennej **musi** być poprzedzone nadaniem jej wartości

Podstawy programowania

spróbujmy:

```
>>> print(fajna_zmienna)
Traceback (most recent call last):
  File "<pyshell#28>", line 1, in <module>
    print(fajna_zmienna)
NameError: name 'fajna_zmienna' is not defined
```

Podstawy programowania

nadanie wartości zmiennej:

- do nadania wartości zmiennej używa się operatora przypisania: =
- nie myl go z operatorem ==
- używa się go tak:

```
zmienna = wyrażenie
```

Podstawy programowania

spróbujmy jeszcze raz:

```
>>> fajna_zmienna=1
>>> print(fajna_zmienna)
1
```

Podstawy programowania

operator przypisania =

- **efekt:** przypisanie wartości wyrażenia z prawej strony operatora do zmiennej wymienionej po lewej stronie

Podstawy programowania

Przypisania można łączyć:

```
a = b = c = 1
```

co należy rozumieć jako ciąg przypisań:

```
a = 1
b = 1
c = 1
```

Podstawy programowania

stąd:

```
>>> a = b = c = 1
>>> print(a,b,c)
1 1 1
```

Podstawy programowania

nadanie wartości zmiennej:

- nadawanie zmiennym sztywnych wartości wprost w kodzie nie zawsze jest dobrym rozwiązaniem
- najczęściej chcielibyśmy podawać pewne wartości dopiero w czasie pracy programu

Podstawy programowania

zauważ:

```
print(a,b,c,d)
```

- jeśli przekazujesz do funkcji więcej argumentów niż jeden, musisz rozdzielać je **przecinkami**

Podstawy programowania

zauważ:

```
print()
```

- jeśli nie przekazujesz do funkcji żadnych argumentów, musisz pozostawić nawiasy

Podstawy programowania

pamiętaj o tym:

```
>>> print
SyntaxError: invalid syntax
```

Podstawy programowania

funkcja input() - wariant #1

- posłuży nam do pobierania danych od użytkownika
- `input()`:
  - argument: żaden
  - efekt: wczytanie wiersza danych z konsoli
  - wynik: napis wprowadzony przez użytkownika
  -
- np:

```
tekst = input()
```

Podstawy programowania

przykład:

```
print("Bądź łaskaw coś napisać i nacisnąć Enter:")
tekst=input()
print("Stało się… Wpisałeś:")
print(tekst)
```

Podstawy programowania

funkcja input() - wariant #2

- posłuży nam do pobierania danych od użytkownika
- `input(x)`:
  - argument: podpowiedź dla użytkownika
  - efekt: wczytanie wiersza danych z konsoli
  - wynik: napis wprowadzony przez użytkownika
  -
- np:

```
tekst = input("Przemów do mnie!")
```

Podstawy programowania

przykład:

```
tekst=input("Bądź łaskaw coś napisać i nacisnąć Enter:")
print("Stało się… Wpisałeś:")
print(tekst)
```

Podstawy programowania

pamiętaj:

- funkcja input() wczytuje **tekst** (ciąg znaków)
- **tekst** nie jest liczbą (nawet jeśli składa się z cyfr)
- jeżeli chcesz użyć wprowadzonego tekstu jako liczby, musisz dokonać jawnej **konwersji** (przekształcenia ciągu znaków na wewnętrzną reprezentację liczby)

Podstawy programowania

w przeciwnym razie zostaniesz upomniany:

```
>>> x=input()
123
>>> y=x/3
Traceback (most recent call last):
  File "<pyshell#6>", line 1, in <module>
    y=x/3
TypeError: unsupported operand type(s) for /:
'str' and 'int'
```

Podstawy programowania

funkcja int()

- konwertuje tekst na liczbę całkowitą
- `int(x)`:
  - argument: tekst reprezentujący liczbę
  - efekt: konwersja tekstu do liczby całkowitej
  - wynik: skonwertowana liczba
  -
- np:

```
ile = int(input())
```

Podstawy programowania

funkcja float()

- konwertuje tekst na liczbę rzeczywistą
- `float(x)`:
  - argument: tekst reprezentujący liczbę
  - efekt: konwersja tekstu do liczby rzeczywistej
  - wynik: skonwertowana liczba
  -
- np:

```
cena = float(input())
```

Podstawy programowania

pamiętaj:

- funkcje int() i float() ufają, że przekazany im argument naprawdę jest zapisem liczby
- jeśli tak nie będzie, funkcje będą zawiedzione

Podstawy programowania

Zostaniesz upomniany:

```
>>> x=int(input())
bulbulator
>>> y=x/3
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    x=int(input())
ValueError: invalid literal for int() with base
10: 'bulbulator'
```

Podstawy programowania

nie trać nadziei:

- istnieje sposób zabezpieczenia się przed nierozważnym działaniem użytkownika
- poznasz go w swoim czasie

Podstawy programowania

A teraz napiszemy program, który podnosi liczbę do kwadratu:

```
liczba=float(input("Podaj liczbę:"))
kwadrat=liczba ** 2
print("Kwadrat z ", liczba, " to ", kwadrat)
```

Podstawy programowania

a teraz to samo, ale z pierwiastkiem

- w Pythonie nie ma operatora, który wykonuje pierwiastkowanie :(
- ale jest funkcja, która umie to zrobić :)
- jednak aby z niej skorzystać, trzeba się trochę postarać :)

Podstawy programowania

funkcja wbudowana

- funkcja, która jest integralną częścią środowiska Pythona
- taką funkcją jest np. print()

Podstawy programowania

moduł:

- moduł to kod, którego nie uruchamia się wprost, a korzysta się z zawartych w nim udogodnień (np. funkcji)
- aby skorzystać z pewnego udogodnienia, należy je **zaimportować** z modułu

Podstawy programowania

- `math`
  moduł zawierający liczne funkcje matematyczne

- `sqrt`
  funkcja z modułu *math*, obliczająca pierwiastek kwadratowy

Podstawy programowania

importowanie – sposób pierwszy

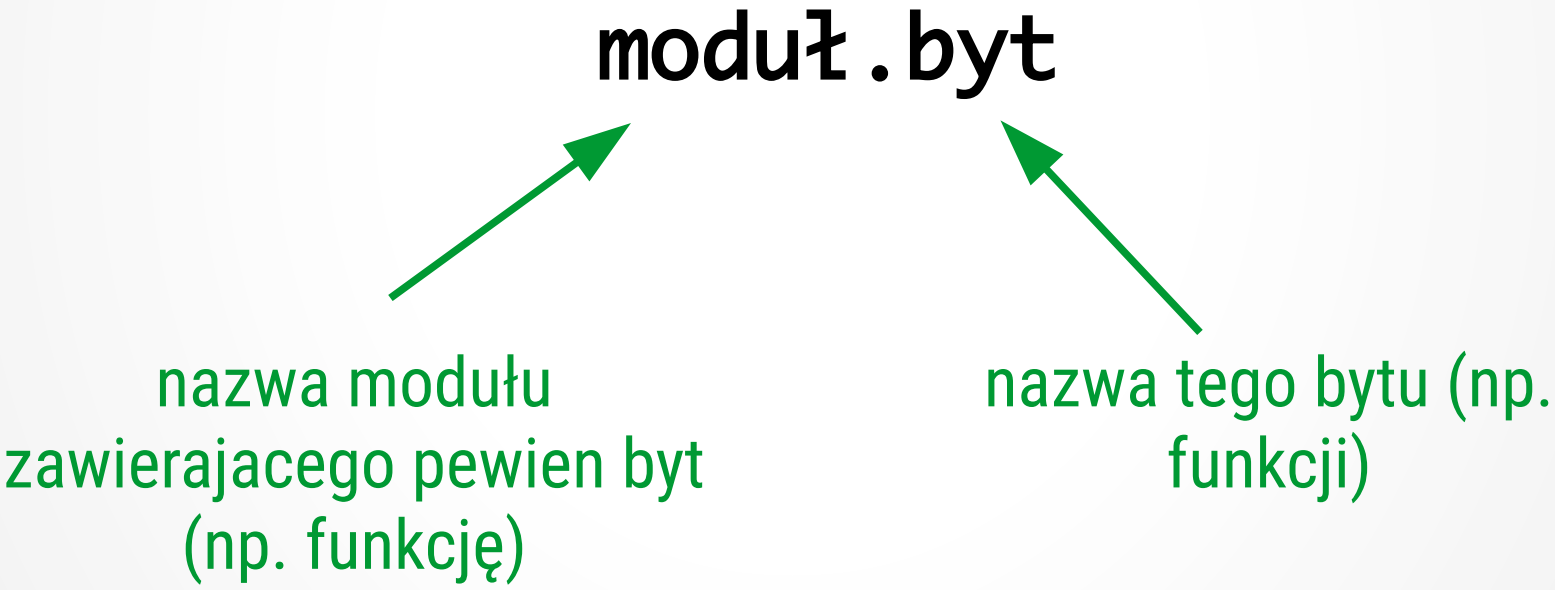
```
import math
```

efekty:

- wszystkie udogodnienia z modułu math stają się dostępne, ale..
- należy je identyfikować tzw. **nazwą kwalifikowaną** np.:
  `math.sqrt`

Podstawy programowania

nazwa kwalifikowana:



Podstawy programowania

Pierwiastkujemy:

```
import math

liczba=float(input("Podaj liczbę:"))
pierwiastek=math.sqrt(liczba)
print("Pierwiastek z ", liczba, " to ", pierwiastek)
```

Podstawy programowania

importowanie – sposób drugi

```
from math import sqrt
```

efekty:

- tylko jawnie wymienione udogodnienia z modułu math stają się dostępne, ale..
- nie trzeba ich identyfikować nazwą kwalifikowaną `math.sqrt`

Podstawy programowania

Pierwiastkujemy po raz drugi:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
pierwiastek=sqrt(liczba)
print("Pierwiastek z ", liczba, " to ", pierwiastek)
```

Podstawy programowania

co się stanie, jeśli spróbujemy pierwiastkować liczbę ujemną?

```
Traceback (most recent call last):
  File "prog.py", line 4, in <module>
    pierwiastek=sqrt(liczba)
ValueError: math domain error
```

Podstawy programowania

trzeba rozgałęzić kod:

- jeśli wartość będzie nieujemna, policzymy pierwiastek
- w przeciwnym przypadku nic nie zrobimy

Podstawy programowania

instrukcja **if** – wariant pierwszy

```
if warunek :
```

warunek:

- wyrażenie logiczne (boolowskie)
- jeśli będzie równe True, instrukcja if uzna, że należy **wykonać** pewne instrukcje
- w przeciwnym wypadku te instrukcje zostaną **pominięte**

Podstawy programowania

zauważ!

- to, które instrukcje stanowią **treść** instrukcji if, w Pythonie jest oznaczane **poziomem wcięcia**!
- wcięcie można uzyskać **spacjami** bądź **tabulacją**
- ten drugi wariant jest **zalecany**
- mieszanie obu wariantów jest **ryzykowne**

Podstawy programowania

Pierwiastkujemy bezpiecznie:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
if liczba >= 0.0:
    pierwiastek=sqrt(liczba)
    print("Pierwiastek z ", liczba, " to ", pierwiastek)
```

Podstawy programowania

zauważ!

- powrót do poprzedniego poziomu wcięcia oznacza koniec instrukcji if

Podstawy programowania

Pierwiastkujemy bezpiecznie:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
if liczba >= 0.0:
    pierwiastek=sqrt(liczba)
    print("Pierwiastek z ", liczba, " to ", pierwiastek)
print("To koniec")
```

Podstawy programowania

instrukcja **if** – wariant drugi

```
if warunek :
        kod1
else :
        kod2
```

- jeśli warunek będzie równy True, instrukcja if uzna, że należy **wykonać** instrukcje stojące za nią
- w przeciwnym wypadku zostaną wykonane instrukcje za else

Podstawy programowania

Pierwiastkujemy jeszcze bezpieczniej:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
if liczba >= 0.0:
    pierwiastek=sqrt(liczba)
    print("Pierwiastek z ", liczba, " to ", pierwiastek)
else:
    print("Podałeś niepoprawną daną")
print("To koniec")
```

Podstawy programowania

instrukcja **if** – wariant trzeci

```
if warunek1 :
        kod1
elif warunek2 :
        kod2
else :
        kod3
```

- jeśli *warunek1* będzie równy True, instrukcja if uzna, że należy wykonać *kod1*
- w przeciwnym wypadku zostanie sprawdzony *warunek2* i jeśli będzie prawdziwy, zostanie wykonany *kod2*
- jeśli i to zawiedzie, zostanie wykonany kod za else (czyli *kod3*)

Podstawy programowania

Pierwiastkujemy tak bezpiecznie, że popadamy w szaleństwo:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
if liczba == 0.0:
    print("E, to wiadomo bez liczenia – zero!")
elif liczba > 0.0:
    pierwiastek=sqrt(liczba)
    print("Pierwiastek z ", liczba, " to ", pierwiastek)
else:
    print("Podałeś niepoprawną daną")
print("To koniec")
```

Podstawy programowania

Zawsze jest więcej niż jeden sposób:

```
from math import sqrt

liczba=float(input("Podaj liczbę:"))
if liczba < 0.0:
    print("Podałeś niepoprawną daną")
elif liczba == 0.0:
    print("Pierwiastek z zera to zero, geniuszu…")
else:
    pierwiastek=sqrt(liczba)
    print("Pierwiastek z ", liczba, " to ", pierwiastek)
print("To koniec")
```

Podstawy programowania

instrukcja **if** – uwagi

- fraza **elif** może wystąpić **wielokrotnie**, ale tylko po **if**
- może również nie wystąpić w ogóle
- fraza **else** może wystąpić co najwyżej jednokrotnie i musi być **ostatnia**
- żadna z tych fraz nie może wystąpić bez wcześniejszego wystąpienia frazy **if**

Podstawy programowania

instrukcja **while** – postać pierwsza

```
while warunek1 :
        kod1
```

- tak długo, jak *warunek1* będzie równy True, instrukcja while będzie wykonywać *kod1*
- jeśli *warunek1* będzie równy False przed przed pierwszym wykonaniem *kod1*, to while zadziała jak if – *kod1* zostanie pominięty

Podstawy programowania

instrukcja **while** – postać druga

```
while warunek1 :
        kod1
else:
        kod2
```

- tak długo, jak *warunek1* będzie równy True, instrukcja while będzie wykonywać *kod1* (być może nie wykona go ani razu)
- jeśli *warunek1* będzie równy False, wykona się *kod2* - conajmniej raz

Podstawy programowania

funkcja **sleep()** z modułu **time**

- zawieszenie programu na wskazaną liczbę sekund
- `sleep(n)`:
  - argument: liczba sekund (jako dana rzeczywista)
  - efekt: odczekanie wskazanej liczby sekund
  - wynik: żaden
  -
- np:

```
time.sleep(3600)
```

← odczeka godzinę

Podstawy programowania

Odliczanie

```
from time import sleep

czas=int(input("Ile sekund odliczyć? "))
while czas > 0:
    print(czas, "...")
    czas = czas - 1
    sleep(1)
print("Do dzieła!")
```

Podstawy programowania

Teraz zmusimy użytkownika do wprowadzenia poprawnej danej

```
from time import sleep

czas=0
while czas <= 0:
    czas=int(input("Ile sekund odliczyć? "))
    if czas <= 0:
        print("Źle! Podaj jeszcze raz!")
while czas > 0:
    print(czas, "...")
    czas = czas - 1
    sleep(1)
print("Do dzieła!")
```

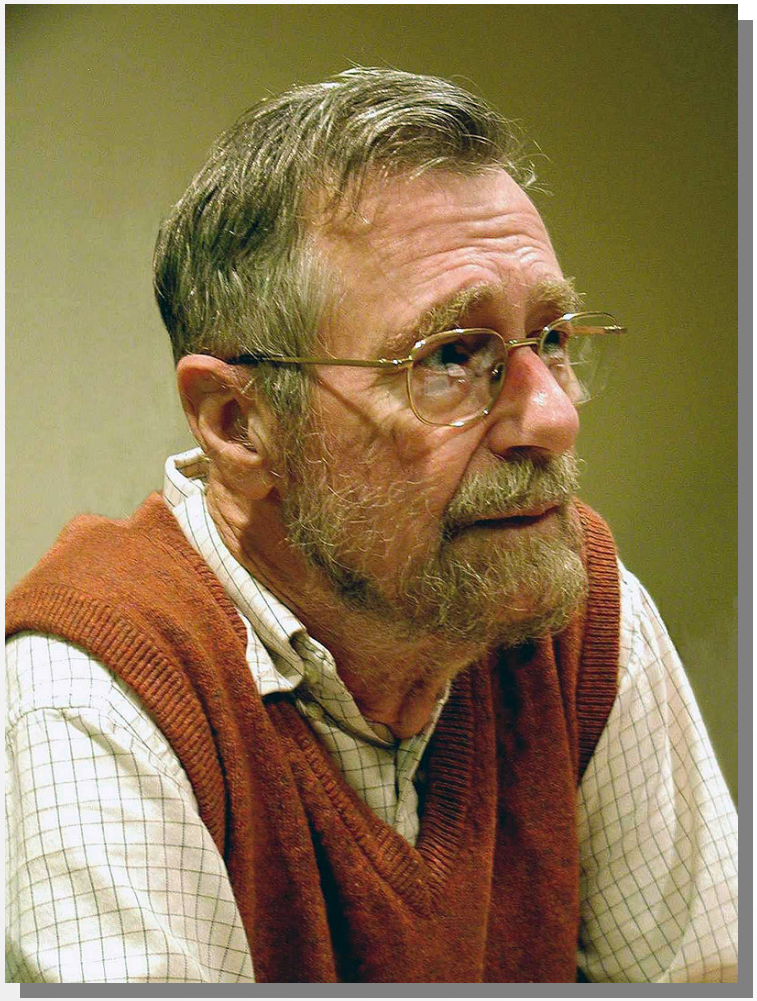
Podstawy programowania

zauważ:

- jeżeli jedna instrukcja if/while jest **zawarta w innej** instrukcji if/while, to manifestujemy to **wzrastającym** zagłębieniem (wcięciem)
- bądź uważny, kiedy posługujesz się wcięciami – złe wcięcia zaowocują **złym zachowaniem kodu**

Podstawy programowania

Za co kochamy Edsgera Dijkstrę?



Edsger Wybe Dijkstra
1930 - 2002

Podstawy programowania

za *Twierdzenie o Strukturze*:

- dowolny algorytm o jednym wejściu i jednym wyjściu można zakodować używając jedynie:
  - złożenia instrukcji
  - instrukcji **if**
  - instrukcji **while**

Podstawy programowania

potrenujmy nieco:

1. program, który wybierze większą z dwóch liczb albo napisze, że obie są równe
2. program, który wybierze największą z trzech liczb
3. program, który wybierze największą z czterech liczb
4. program, który wybierze największą z dowolnej liczby liczb

Podstawy programowania

Program #1:

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
if a == b:
    print("Liczby są równe")
else:
    if a > b:
        print("Większa jest liczba", a)
    else:
        print("Większa jest liczba", b)
```

Podstawy programowania

Program #2:

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
c = int(input("Podaj trzecią liczbę: "))
if a > b:
        if c > a:
            print("Największa liczba to:", c)
        else:
            print("Największa liczba to:", a)
else:
        if c > b:
            print("Największa liczba to:", c)
        else:
            print("Największa liczba to:", b)
```

Podstawy programowania

magiczna sztuczka z funkcją print()

- sekwencja:

```
print("Ala")
print("ma")
print("kota")
```

wyprowadzi na ekran:

```
Ala
ma
kota
```

Podstawy programowania

ale sekwencja:

```
print("Ala", end=" ")
print("ma", end=" ")
print("kota")
```

wyprowadzi na ekran:

```
Ala ma kota
```

Podstawy programowania

a sekwencja:

```
print("Ala", end="")
print("ma", end="")
print("kota")
```

wyprowadzi na ekran:

```
Alamakota
```

Podstawy programowania

- Program #2 poprawiony:

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
c = int(input("Podaj trzecią liczbę: "))
print("Największa liczba to: ", end="")
if a > b:
        if c > a:
            print(c)
        else:
            print(a)
else:
        if c > b:
            print(c)
        else:
            print(b)
```

Podstawy programowania

zalety:

- krótszy kod
- łatwiejsza modyfikacja

pamiętaj:

- jeśli w kilku miejscach robisz to samo, to zastanów się, jak zrobić to tylko w jednym miejscu!

Podstawy programowania

Bądź rzetelny i spolegliwy (*)

- jeśli twój program może wykonać się na kilka sposobów (ma kilka ścieżek wykonania), to sprawdź, jak zachowuje się w każdej z nich
- innymi słowy, tak dobierz dane sprawdzające, aby wykonać wszystkie instrukcje w twoim kodzie
- w ten sposób przeprowadzisz rzetelne **testowanie** swojego kodu

(*) w znaczeniu, w jakim stworzył je prof. Kotarbiński, a nie w jakim używają go politycy

Podstawy programowania

- Program #3

```
a = int(input("Podaj pierwszą liczbę: "))
b = int(input("Podaj drugą liczbę: "))
c = int(input("Podaj trzecią liczbę: "))
d = int(input("Podaj czwartą liczbę: "))
max = a
if b > max:
   max = b
if c > max:
   max = c
if d > max:
   max = d
print("Największa liczba to: ", max, end="")
```

Podstawy programowania

- Program #4 – wariant z liczbą liczb:

```
ile = int(input("Ile podasz mi liczb? "))
max = -999999999
while ile > 0:
    a = int(input("Podaj liczbę: "))
    ile = ile - 1
    if a > max:
        max = a
print("Największą liczba to ", max)
```

Podstawy programowania

- Program #4 – wariant ze „strażnikiem“:

```
max = -999999999
a = 1
licz  = 0
while a != 0:
    a = int(input("Liczba albo 0 aby zakończyć: "))
    if a != 0:
        if a > max:
            max = a
        licz = licz + 1
if licz > 0:
    print("Największa liczba to ", max)
else:
    print("Nie podałeś żadnych liczb :(")
```

Podstawy programowania

sterowanie wykonaniem pętli:

- `break`
  jeśli chcesz opuścić pętlę wcześniej
- `continue`
  jeśli chcesz wcześniej rozpocząć następny obrót pętli

Podstawy programowania

- Program #4 – break:

```
max = -999999999
licz  = 0
while True:
    a = int(input("Liczba albo 0 aby zakończyć: "))
    if a == 0:
        break
    if a > max:
        max = a
    licz = licz + 1
if licz > 0:
    print("Największa liczba to ", max)
else:
    print("Nie podałeś żadnych liczb :(")
```

Podstawy programowania

pętla **for** – pierwsza postać:

```
for x in range(min,max):
    kod
```

- jeśli z góry wiesz, ile razy będziesz chciał wykonać pętlę
- potrzebujesz zmiennej, która będzie liczyć za ciebie
- ale uwaga – tu się kryje pewna pułapka… jaka?

Podstawy programowania

```
for x in range(min,max):
```

tzw. **zmienna sterująca** – przybiera kolejne wartości w kolejnych obiegach pętli; po zakończeniu pętli ma nadal ostatnią użytą wartość

funkcja tworząca zakres (listę) o krańcach określonych parametrami

kraniec **dolny**

kraniec **górny + 1**

Podstawy programowania

zapamiętaj:

```
range(0,max)
```

możesz zapisać krócej jako:

```
range(max)
```

Podstawy programowania

pętla **for** – druga postać:

- ```
  for x in range(min,max):
      kod1
  else:
      kod2
  ```
  - *kod2* wykona się, gdy wartości za frazą in ulegną wyczerpaniu
  ```

Podstawy programowania

- for:

```
for licz in range(0,5):
   print(licz, end=" ")
```

```
0 1 2 3 4
```

Podstawy programowania

- for:

```
for licz in range(0,5):
   print(licz, end=" ")
else:
   print("!")
```

```
0 1 2 3 4 !
```

Podstawy programowania

```
for x in reversed(range(min,max)):
```

Podstawy programowania

- for:

```
for licz in reversed(range(0,5)):
   print(licz, end=" ")
```

```
4 3 2 1 0
```

Podstawy programowania

- I na koniec zagadka – co to jest?

```
for w in range(3):
    x = 20
    s = 1
    for l in range(5):
        for sp in range(x):
            print(end=" ");
        for gw in range(s):
            print("*", end="");
        print();
        x = x - 1
        s = s + 2
```

Podstawy programowania

:)