Notatki python

To trzeba wpisać w powershell żeby można było coś pisać, w IDLE jest wpisane z automatu.

python  
Python 3.6.0 (…)  
Type „help”, „copyright”, „credits” or „license” for more information.

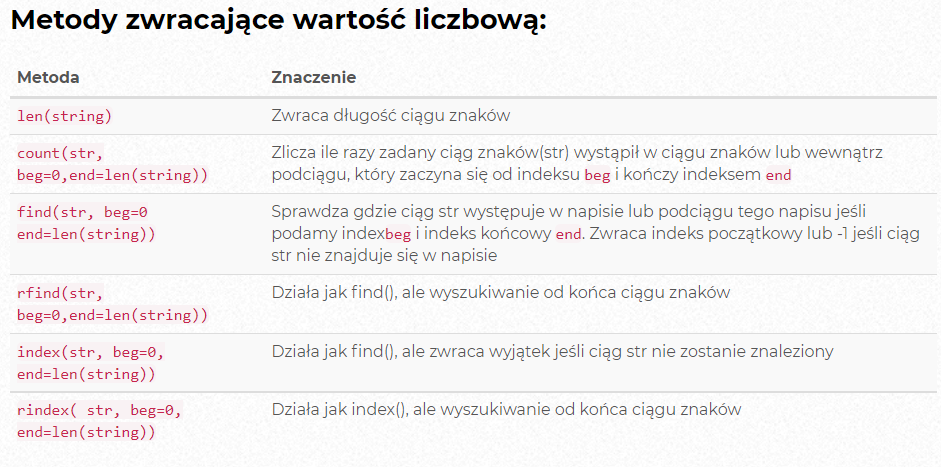
Liczby (Numbers)

Int() Integer liczba całkowita  
Float () zmiennoprzecinkowa  
x\*\*2 to znaczy x2x % y daje resztę z dzielenia x przez   
str(x) zamian zmiennej x integer na zmienna string  
print("Twoje BMI wynosi %f." % BMI) zaokrąglenia liczby w BMI do float – sześć miejsc po przecinku   
print("Twoje BMI wynosi {:f}".format(BMI)) zaokrąglenie liczby w BMI do float – sześć miejsc po przecinku

Napisy (String)

'doesn\'t' za pomocą \ oznaczamy, że ` jest znakiem specjalnym  
int(y) zamiana zmiennej y string na zmienna integer   
\n znak nowej linii, dodanie „entera”  
\t dodanie tabulacji  
\' cudzysłów  
\\ ukośnik  
r stringi mogą być raw – nie wyswietla wtedy znaków specjalnych  
txt[1] można wywoływać elementu stringu – tak jak w listach i krotkach  
txt[-1]możemy cofać się po elementach napisu. Elementem o indeksie [-1] będzie ostatni znak w łańcuchu.  
  
txt[2:] wyświetli od 3 znaku i całą resztę  
txt[1:4] string slicing, wycina od 4 do 10 znaku ale bez znaku nr 10  
txt[:2] zwraca od początku do pozycji 2 ale bez znaku nr 2

  
"\*".join("abrakadabra")  
a\*b\*r\*a\*k\*a\*d\*a\*b\*r\*a



Formatowanie stringów

%  
%s formatowanie stringów  
%d formatowanie liczb – tutaj double  
%f formatowanie liczb – tutaj float  
%r wyświetli to, co zawiera zmienna (raw)  
%.2f formatowanie liczb – tutaj float z zaokrągleniem do 2 miejsc po przecinku  
**%6.3f** – przeznacz 6 znaków na zapisanie liczby zmiennoprzecinkowej, z czego 3 będą po kropce  
**%16s** – przeznacz 16 znaków na zapisanie string’a  
print("1 po angielsku: %s :" % ('one'))

{} formatowanie stringów, double, float – wszystko można wstawić   
{:.2f} formatowanie liczb – tutaj float z zaokrągleniem do 2 miejsc po przecinku  
**{6.3f}** – przeznacz 6 znaków na zapisanie liczby zmiennoprzecinkowej, z czego 3 będą po kropce  
**{16s}** – przeznacz 16 znaków na zapisanie string’a  
"{1} ma {0}".format("Ala", "kota") można wyświetlać w róźnej kolejności  
print("1 po angielsku: {}".format('one'))

Funkcje

Input() wprowadzone dane zapisuje jako string, żeby zapisać jako Float lub Double lub Integer liczba = float(input())  
skrócone używanie: imie = input("Jak masz na imię?")  
lub print(„ile ważysz?”)  
waga = weight(input())  
Print() wyświetla napis  
Help() np. Help(print)   
round(x,5) zaokrąglenie do 5 miejsc po przecinku  
**len()** mierzy ile znaków jest w stringu

# -\*- coding: utf-8 -\*- należy dodać, żeby móc kodować polskie znaki

Listy

**Listy** przechowują różne element poprzedzielane przycinkiem umieszczone w nawiasie kwadartowym. Elementy na liście mogą mieć różne typy.  
lista = [3,5,"herbata", False]  
[3,5,’herbata', False]  
Lista[1] = herbata Można wywoływać według kolejności   
Lista[1] = 4 Można podmieniać wartości

**Krotki** to listy, które są **niezmienne**. Nie można zmodyfikować elementów zawartych w krotce. Krotka, tak jak stringi jest niemutowalna. Krotki umieszczamy w nawiasach okrągłych.  
krotka = (3,5,"kawa", True)  
(3,5,’kawa', True)  
krotka[1] = kawa Można wywował po kolejności ale nie można zmienić wartości!

Słowniki

Przechowuje dane w parach jako klucz : wartość , z czego klucz musi być niemutowalnego typu oraz unikalny w obrębie słownika (klucz nie może się powtarzać! Słownik jest strukturą nieuporządkowaną, w związku z czym nie można odwołać się do elementów słownika po indeksie.   
slownik = { "kawa": "czarna", "herbata" : "zielona", "cukier kostka": 4, 0: "zero"}  
słownik["herbata"]   
zielona   
slownik[1] = "jeden" Można powiększyć słownik dodając niestniejący klucz i przypisując mu wartość  
slownik.keys() – zwraca klucze w słowniku  
slownik.values() – zwraca wartości w słowniku  
list.slownik.keys() – zwróci klucze w formie listy   
"herbata" in slownik – zwróci prawdę jeśli klucz występuje w słowniku.

Instrukcje warunkowe

Porównania:  
x > y zwraca True jeśli x jest większe od y  
x >= y  
x <= y   
x == y zwraca True jeśli x jest równe y  
x != y zwraca True jeśli z nie jest równe y   
and – oznaczającego, że oba warunki muszą być prawdziwe  
or – oznaczającego, że jednen z dwóch warunków musi być spełniony  
x is 10 zwraca True jeśli x równa się 10   
x is not 13 zwraca True jeśli x nie równa się 13  
x is '10' zwraca True jeśli x to na napis „10”  
x in [ 1, 12, 10 ] zwraca True jeśli x zawiera się w tej liście   
x not in [ 1, 12, ‘txt’ ] zwraca True jeśli x nie zawiera się w tej liście   
**Znak == sprawdza czy po obu stronach równania mamy takie same wartości, natomiast is sprawdza czy dwie strony równania są identyczne – dwie zmienne wskazują na ten sam obiekt.  
Przykład:** 'aa' == x \* 2 True  
'aa' is x \* 2 False

**if** ( warunek ):  
    polecenie1  
    polecenie2  
**else**:  
    polecenie3  
    polecenie4

W Pythonie przyjmuje się, że wcięcie powinno składać się z 4 spacji. Dopuszcza się również użycia tabulacji jako wcięcia informacji o bloku kodu.  
Różnica między if i elif: Kolejny if sprawdzi warunek każdorazowo, natomiast elif – kod wykona się, gdy warunek z if jest wyrażeniem falszywym, a elif (warunek) będzie prawdziwy.

Pętla for

**range(start, stop, step)**

Funkcja range() może być używana na 3 sposoby.:

Najbardziej podstawowy i najczęściej używany range(0, end) utworzy sekwencję od 0 do podanej liczby. Zapis ten możemy skrócić. Jeśli w range podamy tylko jedną wartość np. range(5) oznacza, że start i step mają wartości domyślne –*start = 0, step = 1*.

Podając start, możemy zdecydować od jakiego indeksu Python rozpocznie pętlę np. range(3, 10) – range(start, end) pozwala określić początek i koniec zakresu.

Natomiast range(start, end, step) dodaje do tego co jaki krok ma się wykonać pętla.

**Utworzona sekwencja nigdy nie zawiera końca zakresu(!)**.

Oznacza to, że range(0, 5) przeczytamy: *jako wygeneruj zakres od 0 do 5, ale na 5 przestań się wykonywać!*.

Może wydawać się, że funkcja range() tworzy nam niewidoczną listę i dla każdego jej elementu wyświetla wartości. Nic bardziej mylnego! Funkcja range() **nie tworzy bezpośrednio listy, zamiast tego zwraca zakres** – osobny typ w Pythonie. Range tworzy elementy sekwencji po jednym na raz, tym samym unikamy niepotrzebnego przechowywania pełnej sekwencji w pamięci. Po prostu wie jaki ma zakres (start, stop, krok) i przy wywołaniu na bieżąco „dobiera” nowy, następujący element.

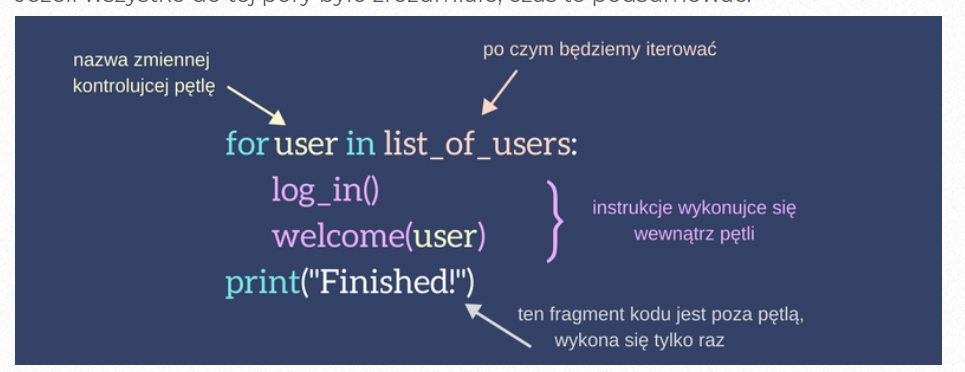
names = ["Ania", "Kasia", "Jan", "Piotr", "Pawel"]  
for i in names:

print("czesc", i)

lub:

for i in range(5):

print("czesc", names[i])



Iterowanie po krotce:

for i in (1,2,3,4):  
    print("krok: ", i)

Iterowanie po stringu:

for i in "ala ma kota":  
    print("krok: ", i)