

Struktura programu

Tabulator = 4 spacje

Dozwolone są spacje pomiędzy operatorami, np.: `bound = x * 12 + 13`, `print (x + 2)`

oraz spacje za przecinkami/średnikami, np.: `print("11", "12")`. Nie wpływają one na działanie kodu, a pozwalają na zachowanie „czystości kodu”

Stałe użycie cudzysłowu, tzn.: `'hello'` lub `"hey"`. Należy zachować jedną konwencję w kodzie.

#komentarz zajmujący 1 linię

"""

Komentarz zajmując wiele linii. Treść zapisana w komentarzu nie jest przetwarzana. Nie wpływa na działanie kodu.

Komentarz wieloliniowy można zapisać również używając pojedynczych cudzysłowów (`'''komentarz'''`)

"""

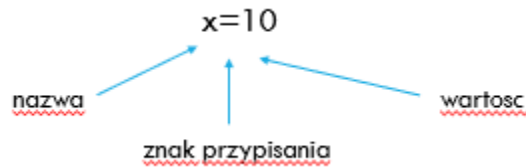
Przykładowy program:

```
print("Ten program dodaje 2 liczby")
num1=input("Wpisz pierwsza liczbę: ")
num1=int(num1)
num2=input("Wpisz druga liczbę: ")
num2=int(num2)
total=num1+num2
print("Suma liczb to " +str(total) +".")
```

Zmienne

Zmienna to miejsce do przechowywania informacji w programie

- każda zmienna posiada **nazwę** oraz **wartość**
- można stworzyć nową zmienną poprzez przypisanie wartości



- można zmienić wartość zmiennej poprzez nowe przypisanie $x=5$
- można ustawić wartość zmiennej za pomocą działań matematycznych $x=5+2$

Przypisanie

Aby przypisać wartość używamy znaku równości (=)

- zmienna jest tworzona przy pierwszym przypisaniu do niej wartości
- kolejne przypisania powodują zmianę wartości zmiennej

UWAGA!!! przypisanie nie oznacza matematycznej równości!!

Przykład:

```
total=total+1
```

- przypisanie: najpierw rozwiązuje prawa stronę, następnie przypisuje wartość do lewej strony

```
total=3
```



```
total=3+1
```
- zmienne są widoczne tylko wewnątrz funkcji, w której są stworzone ("scope")
 - Przykład: jest zmienna stworzona w funkcji `main()`, jest ona widoczna tylko w funkcji `main()`

Nazwy zmiennych

zmienna **MUSI** mieć nazwę i spełniać następujące warunki:

- zaczynać się literą lub od podkreślenia (`_`)
- nie może być wbudowana komenda Pythona (np.: `for`, `if`)
- nie powinna być użyta wbudowana funkcja (np.: `int`, `float`) [miękką zasadą]
- może zawierać tylko litery, cyfry lub podkreślenia

UWAGA!!! ma znaczenie czy nazwa zaczyna się z malej czy z wielkiej litery

- **Zmienna** oraz **zmienna** to dwie różne nazwy

zmienne powinny:

- być opisowe (x nie jest dobrą nazwą, chyba że podajemy koordynaty geograficzne)
- być w formacie snake case: `jakas_zmienna`

Typy zmiennych

Kiedy przechowujemy jaką informację w zmiennej, staje się ona obiektem Pythona

Obiekty mogą być różnych typów i rozmiarów

integer – liczby całkowite, np.: 400, 2, -5	int
float – liczby rzeczywiste, np.: 400.0, 2.0, -5.0	float
string – dane tekstowe, np.: "hello", '10'	str
boolean – wartości logiczne (True/False)	bool

Zmienna typu float nie ma jasno zdefiniowanej „następnej liczby”, np.: po 1.5 może być 1.51 lub 1.501 lub 1.5001, itd...

Zmienna typu int ma zdefiniowaną kolejność liczb, np.: 1, 2, 3...

print()

Funkcja print drukuje tekst w terminalu.

Można użyć kilku komend print, żeby wydrukować kilka linii tekstu:

```
print(„jeden”)                                print(„jeden”, „dwa”, „trzy”)
print(„dwa”)
print(„trzy”)
...
...
print(„n”)
```

Można użyć kilku wersji printa:

```
liczba_rzeczywista = 17.3
liczba_calkowita = 580
tekst = 'chomik'

print(tekst, 'ma', liczba_rzeczywista, 'lat i waży', liczba_calkowita, 'gram') #1

print(tekst + 'ma' + str(liczba_rzeczywista) + 'lat i waży' + str(liczba_calkowita) + 'gram') #2
print(tekst + ' ma ' + str(liczba_rzeczywista) + ' lat i waży ' + str(liczba_calkowita) + ' gram ') #3

print(f'{tekst} ma {liczba_rzeczywista} lat i waży {liczba_calkowita} gram') #4
print(f'{tekst} ma {liczba_rzeczywista} lat i waży {liczba_calkowita} gram') #5
```

outputy:

1 chomik ma 17.3 lat i waży 580 gram

2 chomikma17.3lat i waży580gram

3 chomik ma 17.3 lat i waży 580 gram

4 chomik ma 17.3 lat i waży 580 gram

5 chomik ma 17.3 lat i waży 580 gram

input()

Funkcja input dostaje dane od użytkownika

Drukuje tekst podany w cudzysłowach `"""` lub `''`

- Czeką na odpowiedź użytkownika
- W tym wypadku input jest przypisany do zmiennej (num1)
- Wprowadzone dane są typu tekstowego nawet jeżeli użytkownik wprowadzi numer

Przykład:

```
num1=input('Wpisz pierwszą liczbę: ')
```

Łączenie zmiennych typu *string*

Operator + powoduje połączenie (concatentation) zmiennych typu string

```
Str1="co"  
Str2=" "  
Str3="slychac"  
Str4=Str1+Str2+Str3  
print(Str4)
```

Jest możliwe łączenie zmiennych tego samego typu, ale nie różnego

- **total** było typu int, dlatego trzeba go zamienić na string
- **UWAGA! Originalna wartość total to wciąż int**

Definiowanie typu zmiennej podczas *input()*

Zmienne można również definiować podczas pytania o input()

```
num1=int(input("Wpisz pierwsza liczbe: "))
```

lub

```
num1=float(input("Wpisz pierwsza liczbe: "))
```

Znaki modyfikacji

\ Wstawiony przed znakiem specjalnym zmienia go w regularny znak

Przykład:

```
print('he\'s a boy')
```

```
print("\"hello\"")
```

\t tabulator;

\n nowa linia;

\r powrót karetki;

\b backspace;

Wyrażenia matematyczne

num1 = 7

num2 = 2

num3 = num1 + num2	+	dodawanie	num3=9
num3 = num1 - num2	-	odejmowanie	num3=5
num3 = num1 * num2	*	mnożenie	num3=14
num3 = num1 / num2	/	dzielenie	num3=3.5
num3 = num1 ** num2	**	potęgowanie	num3=49
num3 = num1 // num2	//	dzielenie (wynik int)	num3=3
num3 = num1 % num2	%	reszta z dzielenia	num3=1
num3= -num1	(-)	liczba przeciwna	num3=-7

Działania matematyczne są wykonywane w następującej kolejności:

- nawiasy okrągłe
- potęgowanie
- mnożenie i dzielenie (ten sam priorytet)
- dodawanie i odejmowanie (ten sam priorytet)

Działania o tym samym priorytecie są wykonywane od lewej do prawej strony

W razie wątpliwości lub błędów w wyniku sprawdź/dodaj nawiasy

Stałe

```
PI = 3.14
```

```
INCH_TO_CM = 2.54
```

Stałe pozwalają na zachowanie “czystości” kodu (są dobrze czytelne)

Opisowe nazwy, w formacie UPPER_SNAKE_CASE

Podobieństwo do zmiennych; nie zmieniają wartości podczas egzekucji kodu

Definiujemy stałe w sposób ogólny, żeby wciąż działały w przypadku, gdy zmienimy wartość stałej

Przykład:

```
INCH_TO_CM = 2.54
```

```
inch = float(input('Podaj liczbę cali'))
```

```
cm = inch * INCH_TO_CM
```

```
print(inch, 'cali to', cm, 'centymetrów')
```

Biblioteki w Pythonie

Zbiór kodów do wielokrotnego wykorzystania

Biblioteki zawierają wbudowane m. in. funkcje, stałe, klasy pozwalające na natychmiastowe zastosowanie bez konieczności wyprowadzania skomplikowanych wzorów

<https://docs.python.org/3.14/library/index.html>

Aby zastosować bibliotekę w kodzie należy ją zaimportować. Bez importu, funkcje z biblioteki nie będą działały.

Implementacja biblioteki w kodzie:

↓ ZASTOSOWANIE ↓

```
import nazwa biblioteki
```

(nazwa_biblioteki.funkcja)

aliasy - nazwy bibliotek można zastąpić inną nazwą w celu optymalizacji kodu, np.:

```
import numpy as np
```

(alias.funkcja)

z bibliotek można również importować pojedyncze elementy:

```
from nazwa biblioteki import element
```

(funkcja)

Biblioteka *math*

```
import math
```

Wbudowane stałe:

math.pi	liczba pi
math.e	liczba eulera

Wbudowane funkcje:

math.sqrt(x)	zwraca pierwiastek kwadratowy liczby x
math.exp(x)	zwraca e^x

`math.log(x)` zwraca logarytm naturalny liczby x

wszystkie funkcje i stałe zawarte w bibliotece math:

<https://docs.python.org/3/library/math.html>

Struktura kodu stosując poznane elementy

`import library_name` biblioteki

`CONSTANTS` stałe

`def main():` funkcje

`zmienne` pozostałe

`komendy`

wartości losowe

Sposób na wygenerowanie losowych wartości

Nie można uzyskać w technice komputerowej prawdziwej losowości, więc używamy liczb pseudolosowych

Aby zastosować wartości losowe, należy zaimportować bibliotekę random:

```
import random
```

Funkcja	Działanie
random.randint(min, max)	Zwraca losowy integer z przedziału od min do max, włącznie
random.random()	Zwraca losowy float z przedziału od 0 do 1
random.uniform(min, max)	Zwraca losowy float z przedziału od min do max
random.seed(x)	Ustawia "seed" losowego generatora liczb jako x

Wyrażenia logiczne

To takie, które są prawdziwe lub fałszywe

Podczas stosowania wyrażeń logicznych używamy ==

wyrażenie zwraca wynik True lub False

UWAGA!!

== to nie to samo co =

True i False są wartościami typu boolean (bool)

Zwróćcie uwagę na wielką literę!!

Operatory porównania

==	porównanie
!=	nierówność
>	większość
<	mniejszość
>=	większe lub równe
<=	mniejsze lub równe
is	jedna wartość jest tym samym co druga
is not	wartość nie jest tym samym co druga

Operatory logiczne

and $x > 2$ and $x < 10$

warunek spełniony tylko wtedy, gdy liczba jest z przedziału od 2 do 10

or $x < 2$ or $x > 10$

warunek spełniony wtedy, gdy liczba jest mniejsza od 2 lub większa od 10

not not($x > y$)

warunek spełniony kiedy stwierdzenie jest prawdziwe, czyli jeśli $x > y$ jest fałszywe

Instrukcje warunkowe

if warunek: <- uwaga na dwukropek

tab komenda

UWAGA!! Zwróćcie uwagę na wcięcie

warunek musi być typu boolean (True/False)

warunek musi być prawdziwy, żeby została wykonana komenda

Przykład:

if x>y:

print('x jest większe od y')

if-else

if warunek: <- uwaga na dwukropek

tab komenda

else: <- uwaga na dwukropek

tab inna komenda

Przykład:

if x>y:

print('x jest większe od y')

else:

print('x nie jest większe od y')

if-elif

Przykład:

if warunek:

if x==5:

tab komenda

print('to jest cyfra 5')

```
elif inny_warunek:
```

```
    tab inna_komenda
```

```
elif jeszcze_inny_warunek:
```

```
    tab następną_komenda
```

```
elif x==3:
```

```
    print ('to jest cyfra 3')
```

```
elif x==9:
```

```
    print ('to jest cyfra 9')
```

if-elif-else

Przykład:

```
if warunek:
```

```
    komenda
```

```
elif inny_warunek:
```

```
    inna_komenda
```

```
else:
```

```
    następną_komenda
```

```
if x < y:
```

```
    print ('x jest mniejsza niż y')
```

```
elif x > y:
```

```
    print ('x jest większa niż y')
```

```
else:
```

```
    print ('x i y są równe ')
```

warunki zagnieżdżone

Przykład:

```
if warunek:
```

```
    komenda
```

```
else:
```

```
    if inny_warunek
```

```
        inna_komenda
```

```
    else:
```

```
        następną_komenda
```

```
if x == y:
```

```
    print ('x i y są równe ')
```

```
else:
```

```
    if x < y:
```

```
        print ('x jest mniejsza niż y')
```

```
    else:
```

```
        print ('x jest większa niż y')
```

pętla *while*

while *warunek*:
 komenda

```
x=0
while True:
    print(x)
    x += 1
```

break

- polecenie *break* powoduje wyjście z pętli nawet, jeśli warunek jest prawdziwy

Przykład:

```
x=0
while x<8:
    print(x)
    break
    x = += 1
```

continue

- przeskakuje iterację jeżeli trafi na określony warunek

Przykład:

```
x = 0
while x < 6:
    x += 1
    if x == 3:
        continue
print(x)
```

operatory przypisania

=	x = 5	x = 5
+=	x += 3	x = x + 3
-=	x -= 3	x = x - 3
*=	x *= 3	x = x * 3
/=	x /= 3	x = x / 3
%=	x %= 3	x = x % 3
//=	x //= 3	x = x // 3
**=	x **= 3	x = x ** 3

pętla for

```
for x in range(0, 10):  
    print(x)
```

różnica między pętlami *while* i *for*

- pętlę *while* używamy kiedy nie wiemy ile potrzeba iteracji do osiągnięcia oczekiwanego wyniku
- pętla *for* służy do wykonywania operacji kiedy znamy określoną liczbę iteracji

pętle zagnieżdżone

```
for x in range (0, 3):  
    for y in range (0, 2):  
        print(x, y)
```

***for* z krokiem**

```
for x in range(0, 10, 2):  
    print(x)
```


ciągi znaków

- Każda wartość typu string to sekwencja/ciąg znaków, np.: `zmienna = 'jakas zmienna'`
- Indeksy ciągów zaczynają się od 0

j	a	k	a	s		z	m	i	e	n	n	a
0	1	2	3	4	5	6	7	8	9	10	11	12

- indeksy podajemy w nawiasach kwadratowych []
- indeksy zawsze muszą być liczbą całkowitą (int)
- można używać indeksów wstecz (-1, -2...)

Przykład:

```
zmienna = 'inna zmienna'
```

```
litera = zmienna[4]
```

```
print(litera)
```

wynikiem zmiennej o nazwie litera będzie "i".

- indeks musi być liczbą mniejszą lub równą ilości znaków w sekwencji
- można zastosować funkcję `len()`, która zwraca długość stringa
- funkcję `len()` można zastosować jako parametr określający miejsce w ciągu

Przykład:

```
zmienna = 'to jest jakiś ciąg znaków'
```

```
ostatnia_litera = zmienna[len(zmienna)-1]
```

```
print(ostatnia_litera)
```

wynik: 'w'

ciągi w pętlach

- można zastosować ciągi znaków zarówno w pętli while I for
- wiele obliczeń polega na przetwarzaniu sekwencji znak po znaku

Przykłady:

Pętla while

```
zmienna='kot'
```

```
indeks = 0
```

```
while indeks < len(zmienna):
```

```
    litera = zmienna[indeks]
```

```
    print(litera)
```

pętla for

```
zmienna='kot'
```

```
for litera in zmienna:
```

```
    print(litera)
```

indeks += 1

wycinki

- możliwe jest wycięcie określonego kawałka tekstu za jednym razem
- w tym celu do indeksu wstawiamy dwukropek

Przykład:

zmienna = 'jakas zmienna'

wycinek1 = zmienna[:5] -> 'jakas'

wycinek2 = zmienna[6:] -> 'zmienna'

wycinek3 = zmienna[3:8] -> 'as zm'

- wycinek jest w granicach od n do m-1

stałość stringów

- nie jest możliwe nadpisanie pojedynczego znaku (elementu) w ciągu (obiekcie)
- ciąg znaków jest niezmienny
- możliwe jest:
 - nadpisanie całej zmiennej (zamiana ciągu na inny)
 - utworzenie nowej zmiennej i manipulacje pomiędzy zmiennymi

Przykład:

zmienna = 'jakas zmienna'

nowa_zmienna = 'inna' + zmienna[6:]

operacje na stringach

- można sprawdzić czy dana litera występuje w ciągu znaków
 - 'f' in 'jakas zmienna' -> rozwiązaniem będzie *bool*

Przykład:

```
if 'b' in 'abrakadabra':  
    print('W słowie "abrakadabra" znajduje się litera "b".')
```

- można porównać ciągi (np.: porządkowanie alfabetyczne)
 - 'słowo' > 'litera' !!! Litery z początku alfabetu mają mniejszą wartość

Przykład:

```
warzywo = input('Podaj warzywo: ')

if warzywo < 'cebula':
    print('Twoje warzywo będzie przed słowem cebula')
elif warzywo > 'cebula':
    print('Twoje warzywo będzie po słowie cebula')
else:
    print('Twoje warzywo to cebula')
```

- można przyrównać 2 ciągi
 - 'słowo' == 'słowo'

Przykład:

```
warzywo = input('Podaj warzywo: ')

if warzywo == 'cebula':
    print('Tak, chodziło o cebulę!')
else:
    print('Nie, chodziło o inne warzywo')
```

Istnieją jednak pewne ograniczenia:

- Wielkie litery będą miały pierwszeństwo przed małymi

operator formatowania

var_int = 42

var_float = 3.8

var_string = 'chomik'

print('Mam %d lat' %var_int)

print('Mój wzrost to %g metra' %var_float)

print('Moje ulubione zwierzę to %s' %var_string)

print('Przez ostatnie %g lata %s zjadł %d marchewki' %(var_float, var_string, var_int))

Listy

- listy są kolejnym przykładem sekwencji

Przykłady list:

[7, 8, 4, 67]

['jakas', 'sobie', 'lista']

['lista', 7, 8.4, [9, 78]] <- lista zagnieżdżona

[] <- pusta lista

- listy można przypisać do zmiennych

UWAGA!! W przeciwieństwie do ciągu znaków listy są zmienne

- tak samo jak przy stringach, indeksy liczymy od 0
- można podmieniać pojedyncze elementy list

Przykład:

```
num_list = [8, 56, 73, 79]
```

```
num_list[2] = 23
```

- indeksy działają tak samo jak przy stringach
- najczęstszą metodą poruszania się po listach jest pętla for
- można odczytać kolejno każdy element listy

Przykład:

```
for element in lista:
```

```
    print(element)
```

Łączenie wielu funkcji

- wykorzystując różne funkcje można przeprowadzić więcej operacji na elementach listy

Przykład:

```
for x in range(len(lista)):
```

```
    lista[x] = lista[x] * 6
```

!!! len() jako zakres nie trzeba -1

operacje na listach

- łączenie

```
x = [1, 2]
```

```
y = [3, 4]
```

```
z = x+y          ->    wynik: [1, 2, 3, 4]
```

- powtarzanie

```
x=[1,2]*3        ->    wynik: [1,2, 1,2, 1,2]
```

wycinki list

- podobnie jak przy stringach, można otrzymać wycinki list

```
lista = [1, 't', 56, 'numer', 4.5]
```

```
lista[1:3]          ->    ['t', 56]
```

```
lista[:2]           ->    [1, 't']
```

```
lista[3:]           ->    ['numer', 4.5]
```

```
lista[:]            ->    [1, 't', 56, 'numer', 4.5]
```

- za pomocą wycinków można zmienić kilka elementów na raz

Przykład:

```
lista = [1, 't', 56, 'numer', 4.5]
```

```
lista[2:4] = ['inne', 'elementy']
```

```
lista = [1, 't', 'inne', 'elementy', 4.5]
```

metody

Opis wszystkich metod, które można zastosować do list jest dostępny w dokumentacji Pythona:

<https://docs.python.org/3/tutorial/datastructures.html>

można wywołać spis metod w terminalu, używając funkcji `dir()`:

```
lista = [1, 'string', False, 4.5]
```

```
print(dir(lista))
```

Kolejną przydatną funkcją jest `help()`, która pokazuje w terminalu jak używać metody:

```
print(help(lista))
```

Żeby użyć wybraną metodę należy podać nazwę listy i po kropce nazwę metody z argumentami.

Poniżej znajdują się najważniejsze metody i przykłady ich użycia.

Dodawanie elementów

append()

`lista.append(element)`

!!! elementem może być string, int, lista...

Przykład:

`warzywa = ['pomidor', 'seler', 'marchew']`

`warzywa.append('ziemniak')`

`print(warzywa)`

wynik: ['pomidor', 'seler', 'marchew', 'ziemniak']

Zwracanie indeksu

index()

`lista.index(element)`

!!! tym razem szukamy konkretnego elementu

Przykład:

`warzywa = ['pomidor', 'seler', 'marchew']`

`jaki_indeks = warzywa.index('seler')`

`print(jaki_indeks)`

wynik: 1

Wstawianie elementów

insert()

`lista.insert(element)`

!!! wstawiamy element do listy (niekoniecznie na końcu)

Przykład:

`warzywa = ['pomidor', 'seler', 'marchew']`

`warzywa.insert(1, 'ziemniak')`

`print(warzywa)`

!!! najpierw podajemy indeks

wynik: ['pomidor', 'ziemniak', 'seler', 'marchew']

Usuwanie elementów

remove()

`lista.remove(element)`

Przykład:

`warzywa = ['pomidor', 'seler', 'marchew']`

`warzywa.remove('seler')`

`print(warzywa)`

wynik: ['pomidor', 'marchew']

Sortowanie elementów

`sort()`

`lista.sort()`

!!! nie podajemy elementów

Przykład:

```
warzywa = ['pomidor', 'seler', 'marchew']
```

```
warzywa.sort()
```

```
print(warzywa)
```

wynik: ['marchew', 'pomidor', 'seler']

Krotka [Tuple]

- kolejna ze zbiorów
- krotki są niezmiennie
- krotki są zbiorami uporządkowanymi
- pozwalają na duplikaty

np.: (1, 2, 1)

```
krotka = (1, 't', 56, 'numer', 4.5)
```

!!nawiasy okrągłe

Tworzenie krotki

- przypisanie na zmienną:

```
krotka = (1, 't', 56, 'numer', 4.5)
```

- używając konstruktora

```
krotka = tuple((1, 't', 56, 'numer', 4.5))
```

Krotka z jednym elementem

- Żeby stworzyć krotkę z tylko jednym elementem, należy po nim dostawić przecinek

Przykład:

```
krotka = (1)
```

wynik: `<class int>`

```
krotka = (1,)
```

wynik: `<class tuple>`

Typy danych

int	krotka_int = (1, 2 ,3)
float	krotka_float = (1.6, 2.3 ,3.9)
boolean	krotka_bool = (True, False, False)
string	krotka_string = ('numer', 'jeden')
mieszane	krotka_mix = (1, 't', 56, 'numer', 4.5) Boolean

Operacje na krotkach

- podobnie jak w przypadku listy indeksy zapisujemy w nawiasach kwadratowych
- indeksy zaczynają się od 0

```
krotka = (1, 'numer')  
print(krotka[1])
```

wynik: **'numer'**

Można stosować również funkcje takie jak:

- `type()`
- `len()`
- indeksy ujemne

oraz operacje na krotkach:

- wycinki

```
krotka[:5]  
krotka[5:]  
krotka[2:5]  
krotka[-3:-2]
```

- sprawdzenie czy element istnieje

```
if 'numer' in krotka:  
    print('istnieje')
```

zmienianie wartości

- ponieważ krotki są niezmiennicze, trzeba sobie poradzić inaczej

Przykład:

```
krotka = (1, 2, 3)  
lista = list(krotka)  
lista[1] = 8  
krotka = tuple(lista)
```

dodawanie wartości

- krotka nie posiada metody append, należy zastosować inne metody

1. Konwersja do listy

Przykład:

```
krotka = (1, 2, 3)
lista = list(krotka)
lista.append(4)
krotka = tuple(lista)
```

2. Dodanie kilku krotek

Przykład:

```
krotka1 = (1, 2, 3)
krotka2 = (4,)
krotka1 += krotka2
```

rozpakowywanie krotki

- tworzenie krotki nazywa się jej pakowaniem
- ale można też przeprowadzić operację w drugą stronę

Przykład:

```
warzywa = ('marchew', 'pomidor', 'cebula')
```

```
(pomaranczowy, czerwony, bialy) = warzywa
```

```
print(pomaranczowy)
print(czerwony)
print(bialy)
```

```
wynik: marchew
wynik: pomidor
wynik: cebula
```

UWAGA!!! Liczba zmiennych musi być taka sama jak liczba wartości w krotce

Asterisk (*)

- Jeżeli liczba wartości w krotce przewyższa liczbę zmiennych, można użyć znaku *
- Może wystąpić tylko jeden asterisk na wyrażenie!!!

Przykład:

```
warzywa = ('marchew', 'pomidor', 'cebula', 'pietruszka', 'kalafior')
```

```
(pomaranczowy, czerwony, *bialy) = warzywa
```

```
print(pomaranczowy)
print(czerwony)
print(bialy)
```

```
wynik: marchew
wynik: pomidor
wynik: ['cebula', 'pietruszka', 'kalafior']
```

krotki w pętlach

```
krotki = ('marchew', 'pomidor', 'cebula')
```

```
for i in krotki:  
    print(i)
```

```
for i in range(len(krotki):  
    print(krotki[i])
```

```
i = 0  
while i < len(krotki):  
    print(krotki[i])  
    i += 1
```

łączenie

```
krotka1 = (1, 2, 3)  
krotka2 = (4, 5, 6)  
krotka3 = krotka1 + krotka2  
print(krotka3)
```

wynik: (1, 2, 3, 4, 5, 6)

powielanie

```
krotka = ('a', 'b', 'c')  
powielona = krotka * 2
```

```
print(powielona)
```

wynik: ('a', 'b', 'c', 'a', 'b', 'c')

metody.. znowu

- count()
zwraca ile razy dana wartość pojawiła się w krotce

```
krotka = (1, 2, 4.5, True, 1, 'a', 6, 'numer', 1)  
licz = krotka.count(1)  
print(licz)
```

wynik: 3

- index()
zwraca indeks, na którym dana wartość pojawiła się w krotce

```
krotka = (1, 2, 4.5, True, 1, 'a', 6, 'numer', 1)  
licz = krotka.index(1)  
print(licz)
```

wynik: 0

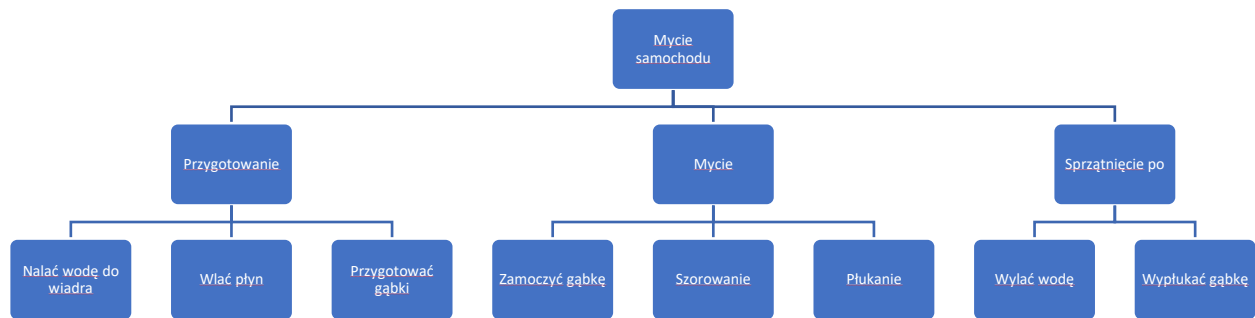
dostęp do elementów zagnieżdżonych

```
krotka1 = (54, [7, 24, 67], (22, 97, 56))
```

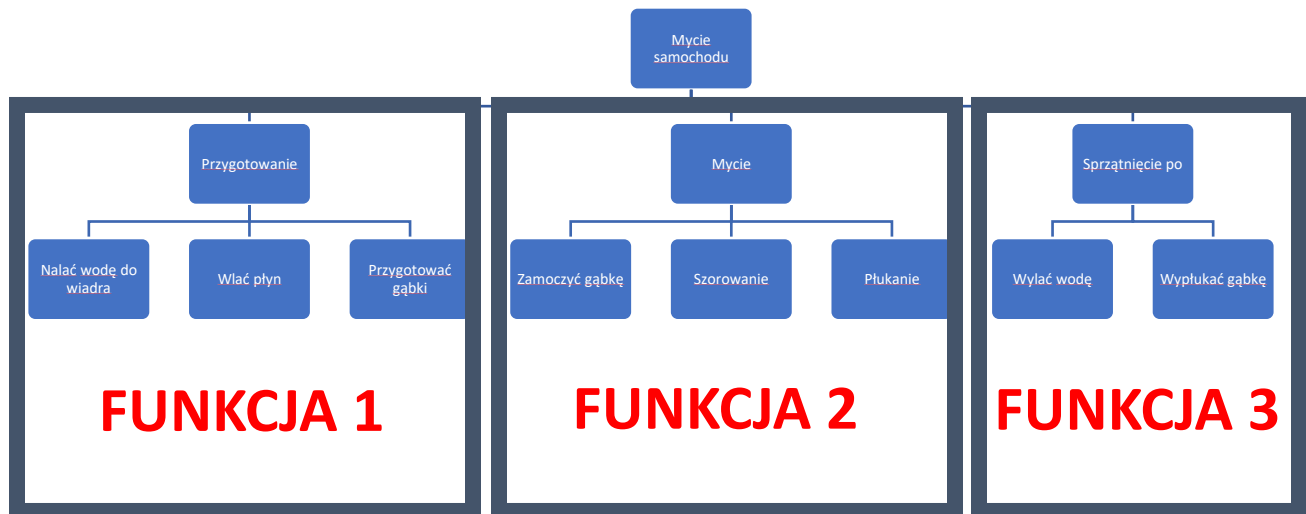
```
print(krotka1[2][0])
```

FUNKCJE

Dekompozycja polega na podzieleniu kodu na jak najmniejsze kroki.



Zbiór kroków nazywamy funkcją



funkcje wbudowane

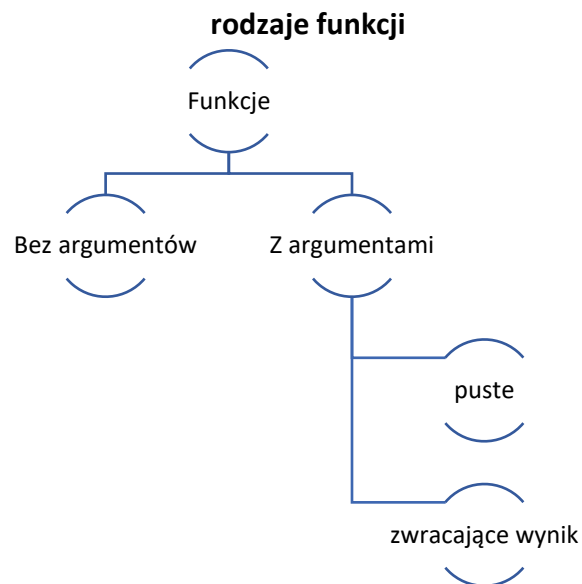
W Pythonie istnieją już funkcje wbudowane, które można tylko wywołać, bez potrzeby pisania ich na nowo.

Pewne funkcje wbudowane były już używane podczas zajęć:

print()
float()
str()
int()
len()
type()

Wszystkie funkcje wbudowane w Pythonie, ich opis oraz sposób wykorzystania można znaleźć na stronie z dokumentacją:

- <https://docs.python.org/3/library/functions.html>



definiowanie funkcji

W Pythonie można zdefiniować własne, dowolne funkcje

```
def nazwa_funkcji():  
    zawartość  
    zawartość  
    zawartość
```

← **UWAGA!! Dwukropek**

← W przypadku funkcji bez argumentów nawias powinien być pusty

Przykład:

```
def wydruk():  
    print('Jest super...')  
    print('...albo i nie')
```

Niektóre funkcje wymagają jednak argumentów (np.: type(zmienna))

Takie funkcje mogą zwracać jakąś wartość/wynik lub nie (funkcja pusta)

Przykład:

```
def dodawanie(x, y):  
    suma = x+y  
    return suma  
  
def main():  
    total = dodawanie(4, 5)  
    print(total)  
  
main()
```

nazwy funkcji

- nazwa funkcji może być dowolnym słowem
- lub ciągiem słów w formacie snake_case (słowa połączone podkreślnikiem)
- tak jak przy zmiennych, nazwa funkcji może zawierać litery, cyfry oraz niektóre znaki interpunkcyjne
- nazwa nie może zaczynać się od cyfry
- nie można używać słowa kluczowego jako nazwy funkcji
- nie powinno się używać tej samej nazwy funkcji i zmiennej

wywoływanie funkcji

Żeby funkcja zadziałała należy ją wywołać

Funkcje zdefiniowane samodzielnie, wywołujemy w taki sam sposób, co funkcje wbudowane

Przykład dla funkcji bez argumentów:

```
def wydruk():  
    print('Jest super...')  
    print('...albo i nie')  
wydruk()
```

Przykład dla funkcji z argumentami:

```
def dodawanie(x, y):  
    suma = x+y  
    return suma  
  
def main():  
    total = dodawanie(4, 5)  
    print(total)  
  
main()
```

<- UWAGA!!! return używamy tylko kiedy oczekujemy zwrotu parametrów z funkcji

Control Flow

W programowaniu ważna jest kolejność wykonywania instrukcji.

Na początku powiedziano, że instrukcje są wykonywane linijka po linijce.

Ale...

Zostały wprowadzone elementy, które powodują obejście niektórych linii lub ich powielanie (if, while, for)

Odnosi się to również do funkcji.

Dlatego też należy znać zasady kontroli sterownia i upewnić się, że instrukcje wykonywane są w odpowiedniej kolejności.

ĆWICZENIA

1. Napisz program, który zapyta użytkownika o imię oraz go powita

Przykładowy output:

Jak masz na imię? Paulina

Witaj, Paulina

2. Napisz program, który rozwiąże w jakim wieku są osoby

Maciek ma 21 lat

Ula jest 6 lat starsza niż Maciek

Czarek jest 20 lat starszy niż Ula

Antek ma tyle lat co Czarek i Maciek razem

Kasia ma tyle samo lat co Czarek

Program powinien wydrukować wiek każdej osoby w oddzielnej linii

3. Napisz program, który zapyta użytkownika jakie jest jego ulubione zwierzę, a następnie odpowie, że to też jego ulubione zwierzę

Przykładowy output:

Jakie jest Twoje ulubione zwierzę? **krowa**

Moje ulubione zwierzę to także krowa

4. Napisz program, który zapyta użytkownika o cyfrę, a następnie wydrukuje drugą potęgę tej liczby

Przykładowy output:

Podaj liczbę: **3**

3 do kwadratu to 9

5. Poproś użytkownika o wprowadzenie długości każdego z boków trójkąta, a następnie wydrukuj wartość obwodu

Przykładowy output:

Jaka jest długość pierwszego boku? **1**

Jaka jest długość pierwszego boku? **2**

Jaka jest długość pierwszego boku? **3**

Odwód trójkąta to **6!**

6. Użytkownik powinien wprowadzić temperaturę w stopniach Celsjusza, a program powinien mu powiedzieć jaka to temperatura w stopniach Farenheita.

Wzór na konwersję:

$$F = C * 9 / 5 + 32$$

Przykładowy output:

Podaj temperaturę [st. C]: **20**

20 stopni Celsjusza to 68 stopni Farenheita

7. Zapytaj użytkownika o podanie 2 liczb, a następnie wyświetl wynik odejmowania drugiej liczby od pierwszej liczby

8. Mad Libs: Napisz program, który zastąpi wyróżnione słowa, używając stałe

Był sobie czarodziej o imieniu (imię), który uwielbiał jeść (owoc).

(imię) zawsze trzymał zapas (liczba) (owoc) w swojej lodówce!

Pewnego dnia, (imię) zdał sobie sprawę, że nie mogą zatrzymać tych wszystkich (owoc) dla siebie, więc sprzedał je na targu po (koszt) za sztukę,

a za zarobione pieniądze kupił owoce do podzielenia się z całą wioską!

Legenda głosi, że (ilość lat) lat później (imię) nadal je owoce.

9. Waga Ziemianina na Marsie wynosi 37,8% ich wagi na Ziemi. Napisz program, który prosi Ziemianina o podanie swojej wagi na Ziemi i wyświetla obliczoną wagę na Marsie.

Przykładowy output:

Podaj wagę na Ziemi: **120**

Ekwiwalent na Marsie: 45.36

10. Oblicz pierwiastek kwadratowy liczby podanej przez użytkownika używając biblioteki math

Przykładowy output:

Podaj liczbę: 3

Pierwiastek kwadratowy liczby 3.0 to 1.7320508075688772

11. Napisz program, który symuluje rzut dwiema kostkami i drukuje wyniki każdego rzutu, jak również sumę.

Przykładowy output:

Każda z kości ma 6 ścianek

Pierwsza kość: 3

Druga kość: 5

Suma kości: 8

12. Napisz program, który wygeneruje 2 losowe liczby z zakresu od 0 do 99 i je dodać. Użytkownik próbuje zgadnąć odpowiedź. Program mu mówi czy zgadł dobrze

Oczekiwany output:

Jaka jest suma 58+45?

Jaka jest suma 58+45?

Twoja odpowiedź: 103

Twoja odpowiedź: 130

Odpowiedź prawidłowa!

Źle! Poprawna odpowiedź to 103

13. Napisz program, który zapyta użytkownika o wiek, a następnie powie mu czy może głosować w 3 fikcyjnych krajach: Etgidi (15), Siqira (57), Gmis (25).

Przykładowy output:

Ile masz lat? **21**

Możesz głosować w Etgidi, gdzie wiek wyborczy to 15

Nie możesz głosować w Siqiri, gdzie wiek wyborczy to 57

Nie możesz głosować w Gmis, gdzie wiek wyborczy to 25

14. Napisz program, który zasymuluje rzut dociążoną monetą. Szansa na wylosowanie orła: 70%

15. Napisz program, który powie użytkownikowi czy podany przez niego rok jest przestępny czy nie.

Aby rok uznać za przestępny muszą być spełnione warunki:

- jest podzielny przez 4 i niepodzielny przez 100 lub
- jest podzielny przez 400

16. Napisz program, który wypisze ciąg Fibbonacciego do określonego limitu (np.: 1000)

17. Napisz program, który pyta użytkownika o liczbę i je dodaje. Program kończy się kiedy użytkownik poda 0.

18. Napisz program, który zasymuluje "magic 8-ball". Użytkownik zadaje pytanie, na które odpowiedź jest tak lub nie. Magiczna kula pokazuje mu odpowiedź.

Możliwe odpowiedzi kuli:

- 1) Tak
- 2) Nie ma mowy
- 3) Tylko kula wie
- 4) Bezwzględnie
- 5) Zapytaj później

Przykładowy output:

Czy dostanę awans?

Tylko kula wie

19. Poproś użytkownika o podanie liczby, następnie napisz program, który policzy od 1 do tej liczby oraz wypisze je. Jeśli liczba jest podzielna przez 3 zastąp ją tekstem "Fizz", jeśli jest podzielna przez 5 – zastąp ją napisem "Buzz", natomiast jeśli jest podzielna przez 3 i 5 zastąp ją tekstem "FizzBuzz". Na koniec podlicz numer "Fizz", "Buzz" i "FizzBuzz"

Oczekiwany output:

1
2
Fizz
4
Buzz

20. Poproś użytkownika o podanie liczby i napisz program, który doda wszystkie liczby od 1 do podanej (np.: 5 to 1+2+3+4+5)

Przykładowy output:

Podaj liczbę: 5
Suma to 15

21. Napisz pętlę while, która zaczyna się od ostatniego znaku w napisie i działa od końca, do pierwszego znaku, wypisując każdą literę w osobnej linii, w odwrotnej kolejności.
22. Napisz program, który policzy ile z każdej litery występuje w słowie “rabarbar”
23. Napisz program, który poprosi użytkownika o 5 słów i powie mu które z nich jest najdłuższe oraz ile ma liter
24. Napisz program, który podniesie każdy z elementów listy do kwadratu
25. Napisz program, który policzy ile jest elementów w liście (bez użycia funkcji len())

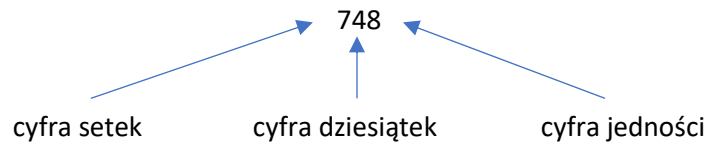
26. Napisz program, który doda pola 3 trójkątów

Trójkąt 1:	Trójkąt 2:	Trójkąt 3:
a=3	a=8	a=7
h=5	h=12	h=4

Oczekiwany output:

Pole wszystkich trójkątów to: 69.5

27. Napisz program, który przyjmuje integer i drukuje jego cyfrę jedności



Przykładowy output:

Podaj liczbę: **748**

Cyfra jedności to 8

28. Zamień krotki:

krotka1 = (0, 45, 67)

krotka2 = (99, 65, 32)

Oczekiwany output:

krotka1: (99, 65, 32)

krotka2: (0, 45, 67)

29. Skopiuj kawałek jednej krotki do innej

krotka1 = (54, 7, 24, 67, 22, 97, 56)

Oczekiwany output:

krotka2 = (67, 22, 97)

ODPOWIEDZI:

1.

```
name = input('Jak masz na imię? ')
print('Witaj,', name)
```

2.

```
maciek = 21
ula = maciek + 6
czarek = ula + 20
antek = czarek + maciek
kasia = czarek
print('Maciek ma ', maciek, 'lat')
print('Ula ma ', ula, 'lat')
print('Czarek ma ', czarek, 'lat')
print('Antek ma ', antek, 'lat')
print('Kasia ma ', kasia, 'lat')
```

3.

```
animal = input('Jakie jest Twoje ulubione zwierzę? ')
print('Moje ulubione zwierzę to też', animal)
```

4.

```
num = int(input('Podaj liczbę: '))
pow = num * num
print(num, 'do kwadratu to', pow)
```

5.

```
num1 = int(input('Jaka jest długość pierwszego boku? '))
num2 = int(input('Jaka jest długość drugiego boku? '))
num3 = int(input('Jaka jest długość trzeciego boku? '))
total = num1 + num2 + num3
print('Obwód trójkąta to ' + str(total) + '!')
```

6.

```
celsius = float(input('Podaj temperaturę [st. C] '))
fahrenheit = celsius * 5/9 + 32
print(str(celsius) + ' stopni Celsjusza to ' + str(fahrenheit) + ' stopni Farenheita')
```

7.

```
print('Ten program podaje różnicę.')
num1=float(input('Podaj pierwszą liczbę: '))
num2=float(input('Podaj drugą liczbę: '))
print('Wynik to ' + str(num1-num2))
```

8.

```
IMIE = 'Merlin'
OWOC = 'mango'
LICZBA = 5873
KOSZT = 43
ILOSC_LAT = 200

print('Był sobie czarodziej o imieniu ' + IMIE + ', który uwielbiał jeść '
      + OWOC + '.')
print(IMIE + ' zawsze trzymał zapas ' + str(LICZBA) + ' ' + OWOC + ' w swojej
      łódźce!')
print('Pewnego dnia ' + IMIE + ' zdał sobie sprawę, że nie może zatrzymać tych
      wszystkich ' + OWOC + ' dla siebie,')
print('więc sprzedał je na targu po ' + str(KOSZT) + ' za sztukę,')
print('a za zarobione pieniądze kupił owoce do podzielenia się z całą
      wioską!')
print('Legenda głosi, że ' + str(ILOSC_LAT) + ' lat później ' + IMIE + ' nadal je
      owoce.')
```

9.

```
MARS_MULTIPLE = 0.378

earth_weight_str = input('Podaj wagę na Ziemi: ')
earth_weight = float(earth_weight_str)
mars_weight = earth_weight * MARS_MULTIPLE
print('Ekwiwalent na Marsie: ' + str(mars_weight))
```

10.

```
import math

num = float(input("Podaj liczbę: "))
root = math.sqrt(num)
print("Pierwiastek kwadratowy liczby ", num, "to", root)
```

11.

```
import random

NUM_SIDES = 6

# random.seed(1)
die1 = random.randint(1, NUM_SIDES)
die2 = random.randint(1, NUM_SIDES)
total = die1 + die2
print("Dice have", NUM_SIDES, "sides each.")
print("First die:", die1)
print("Second die:", die2)
print("Total of two dice:", total)
```

12.

```
import random
MIN RAND=0
MAX RAND=99

num1=random.randint(MIN RAND,MAX RAND)
num2=random.randint(MIN RAND,MAX RAND)
total=num1+num2
print('Jaka jest suma ' +str(num1) + ' + ' +str(num2) +'?')
answer=int(input('Twoja odpowiedź: '))
if answer==total:
    print('Odpowiedź prawidłowa!')
else:
    print('Źle! poprawna odpowiedz to ' +str(total))
```


13.

```
ETGIDI=15
SIQIRA=57
GMIS=25

age=int(input('Ile masz lat? '))
if age>ETGIDI:
    print('Możesz głosować w Etgidi, gdzie wiek wyborczy to ' +str(ETGIDI)
+'.')
else:
    print('Nie możesz głosować w Etgidi, gdzie wiek wyborczy to
'+str(ETGIDI) +'.')
if age>SIQIRA:
    print('Możesz głosować w Siqiri, gdzie wiek wyborczy to '+str(SIQIRA)
+'.')
else:
    print('Nie możesz głosować w Siqiri, gdzie wiek wyborczy to
'+str(SIQIRA) +'.')
if age>GMIS:
    print('Możesz głosować w Gmis, gdzie wiek wyborczy to '+str(GMIS) +'.')
else:
    print('Nie możesz głosować w Gmis, gdzie wiek wyborczy to '+str(GMIS)
+'.')
```

14.

```
import random

ORZEL=0.7

if random.random()<ORZEL:
    print("Orzeł")
else:
    print("Reszka")
```

15.

```
year=int(input('Podaj rok. '))
if year%4==0:
    if year%100==0:
        if year%400==0:
            print("To jest rok przestępny")
        else:
            print("To nie jest rok przestępny")
    else:
        print("To jest rok przestępny")
else:
    print("To nie jest rok przestępny")
```

16.

```
LIMIT = 10000
fib1 = 0
fib2 = 1

while fib1 <= LIMIT:
    print(fib1)
    fib_next = fib1 + fib2
    fib1 = fib2
    fib2 = fib_next
```

17.

```
num = int(input("Podaj liczbę: "))
suma = num

while num != 0:
    print("Suma do tej pory to: " + str(suma))
    num = int(input("Podaj liczbę: "))
    suma = suma + num
```

18.

```
import random

ODP_1 = "Tak!"
ODP_2 = "Nie ma mowy."
ODP_3 = "Tylko kula wie."
ODP_4 = "Bezwzględnie."
ODP_5 = "Zapytaj później"

pytanie=input('Zadaj pytanie tak/nie ')

while pytanie!="":

    nr_odp = random.randint(1, 5)
    if nr_odp == 1:
        print(ODP_1)
    if nr_odp == 2:
        print(ODP_2)
    if nr_odp == 3:
        print(ODP_3)
    if nr_odp == 4:
        print(ODP_4)
    if nr_odp == 5:
        print(ODP_5)
    pytanie=input('Zadaj pytanie tak/nie ')
```

19.

```
count_fizz=0
count_buzz=0
count_fizzbuzz=0

num=int(input('Podaj liczbę: '))
for i in range(1,num+1,1):
    if i%5==0 and i%3==0:
        count_fizzbuzz+=1
        print('Fizzbuzz')
    elif i%3==0:
        count_fizz+=1
        print('Fizz')
    elif i%5==0:
        count_buzz+=1
        print('Buzz')
    else:
        print(i)

print()
print('licznik Fizz: ' +str(count_fizz))
print('licznik Buzz: ' +str(count_buzz))
print('licznik FizzBuzz: ' +str(count_fizzbuzz))
```

20.

```
num = int(input('Podaj liczbe: '))
suma=0
for x in range(1, num+1):
    suma+=x
print('Suma to ' +str(suma))
```

21.

```
zmienna = 'jakas zmienna'
indeks=len(zmienna)-1
while indeks >= 0:
    litera = zmienna[indeks]
    print(litera)
    indeks-=1
```

22.

```
slovo = 'rabarbar'
licznik_r = 0
licznik_a = 0
licznik_b = 0

for litera in slovo:
    if litera=='r':
        licznik_r+=1
    elif litera=='a':
        licznik_a+=1
    else:
        licznik_b+=1

print('W slowie "rabarbar" występuja ' +str(licznik_r) +' litery "r"')
print('W slowie "rabarbar" występuja ' +str(licznik_a) +' litery "a"')
print('W slowie "rabarbar" występuja ' +str(licznik_b) +' litery "b"')
```

23.

```
length = 0
for x in range (5):
    word = input('Podaj słowo: ')
    new_len = len(word)

    if new_len>length:
        longest_word = word
        length = len(longest_word)

print('Najdłuższe słowo to ' +longest_word +', a jego długość to ' +str(length))
```

24.

```
list = [4, 6, 2, 8, 1]

for num in range(len(list)):
    squared = list[num]**2
    print(squared)
```

25.

```
list = [7, 'samochod', [3, 6], 'omega', 5.2, 154]
counter = 0

for element in list:
    counter+=1

print('Liczba elementów w liście to: ' +str(counter))
```

26.

```
def pole_trojkata(a, h):
    pole=0.5*a*h
    return pole

def main():
    trojkat_1 = pole_trojkata(3, 5)
    trojkat_2 = pole_trojkata(8, 12)
    trojkat_3 = pole_trojkata(7, 4)

    suma = trojkat_1 + trojkat_2 + trojkat_3

    print(suma)

main()
```

27.

```
def jednosci(num):
    cyfra = num%10
    print('Cyfra jedności to 'cyfra)

def main():
    num = int(input('Podaj liczbę: '))
    jednosci(num)

main()
```

28.

```
krotka1 = (0, 45, 67)
krotka2 = (99, 65, 32)

krotka1, krotka2 = krotka2, krotka1

print(krotka1)
print(krotka2)
```

29.

```
krotka1 = (54, 7, 24, 67, 22, 97, 56)
krotka2 = krotka1[3:-1]

print(krotka2)
```