



support

**Zeszyt ćwiczeń**  
**Programowanie Python - składnia**  
**(poziom podstawowy)**



MINISTERSTWO  
ŚRODOWISKA



Dofinansowano ze środków  
Narodowego Funduszu  
Ochrony Środowiska  
i Gospodarki Wodnej

<b>Ćwiczenie 1. Uruchomienie Pythona 3 w konsoli OSGeo4W Shell</b>	<b>5</b>
Treść zadania	5
Opis	5
<b>Ćwiczenie 2. Praca bezpośrednio w interpreterze Pythona</b>	<b>6</b>
<b>Ćwiczenie 3. Stworzenie i uruchomienie skryptu</b>	<b>7</b>
Treść	7
Opis	7
<b>Ćwiczenie 4. Import obiektów</b>	<b>7</b>
Treść zadania	7
Kod źródłowy	8
<b>Ćwiczenie 5. Instalacja nowej biblioteki</b>	<b>8</b>
Treść zadania	8
Opis	8
<b>Ćwiczenie 6. Odległość pomiędzy punktami</b>	<b>9</b>
Treść zadania	9
Opis algorytmu	9
Kod źródłowy	10
<b>Ćwiczenie 7. Sprawdzenie czy losowa liczba jest liczbą pierwszą</b>	<b>11</b>
Treść zadania	11
Opis	11
Kod źródłowy	11
<b>Ćwiczenie 8. Znajdowanie liczb pierwszych w danym zasięgu</b>	<b>12</b>
Treść zadania	12
Opis zadania	12
Kod źródłowy	12
<b>Ćwiczenie 9. Formatowanie tekstu</b>	<b>12</b>
Treść zadania	12
Opis zadania	12
Kod źródłowy	12
<b>Ćwiczenie 10. Długość linii</b>	<b>13</b>
Treść zadania	13
Opis algorytmu	13
Kod źródłowy	14
<b>Ćwiczenie 11. Generowania słownika z liczb</b>	<b>15</b>
Treść zadania	15
Opis	15
Kod źródłowy	15
<b>Ćwiczenie 12. Średnia geometryczna elementów listy</b>	<b>15</b>
Treść zadania	15
Opis	15

Kod źródłowy	16
<b>Ćwiczenie 13. Totolotek</b>	<b>16</b>
Treść zadania	16
Opis	16
Kod źródłowy	17
<b>Ćwiczenie 14. Funkcja do obliczania obwodu i pola koła</b>	<b>18</b>
Treść zadania	18
Kod źródłowy	18
<b>Ćwiczenie 15. Funkcje do obliczania odległości i długości</b>	<b>19</b>
Treść zadania	19
Opis	19
Kod źródłowy	19
<b>Ćwiczenie 16. Klasy punktów i linii</b>	<b>20</b>
Treść zadania	20
Opis algorytmu	20
Kod źródłowy	21
<b>Ćwiczenie 17. Dziedziczenie klas</b>	<b>22</b>
Treść zadania	22
Opis	22
Kod źródłowy	22
<b>Ćwiczenie 18. Zapisywanie danych do plików</b>	<b>23</b>
Treść zadania	23
Opis	23
Kod źródłowy	24
<b>Ćwiczenia do samodzielnego wykonania</b>	<b>25</b>
Ćwiczenie 19. Średnia arytmetyczna	25
Treść zadania	25
Wynik	25
Ćwiczenie 20. Modyfikacja tekstu	25
Treść zadania	25
Wynik	25
Ćwiczenie 21. Wyszukiwanie elementów list	25
Treść zadania	25
Wynik	25
Ćwiczenie 22. Usuwanie duplikatów z kolekcji	26
Treść zadania	26
Wynik	26
Ćwiczenie 23. Tekst na słownik	26
Treść zadania	26
Wynik	26

Ćwiczenie 24. Przeliczanie stopni	26
Treść zadania	26
Wynik	26
Ćwiczenie 25. Wyszukiwanie dzielników liczby	27
Treść zadania	27
Wynik	27
Ćwiczenie 26. Ciąg Fibonacciego	27
Treść zadania	27
Wynik	27
Ćwiczenie 27. Parser plików CSV	28
Treść zadania	28
Wynik	28
Ćwiczenie 28. Zapis listy obiektów do pliku	28
Treść zadania	28
Wynik	28

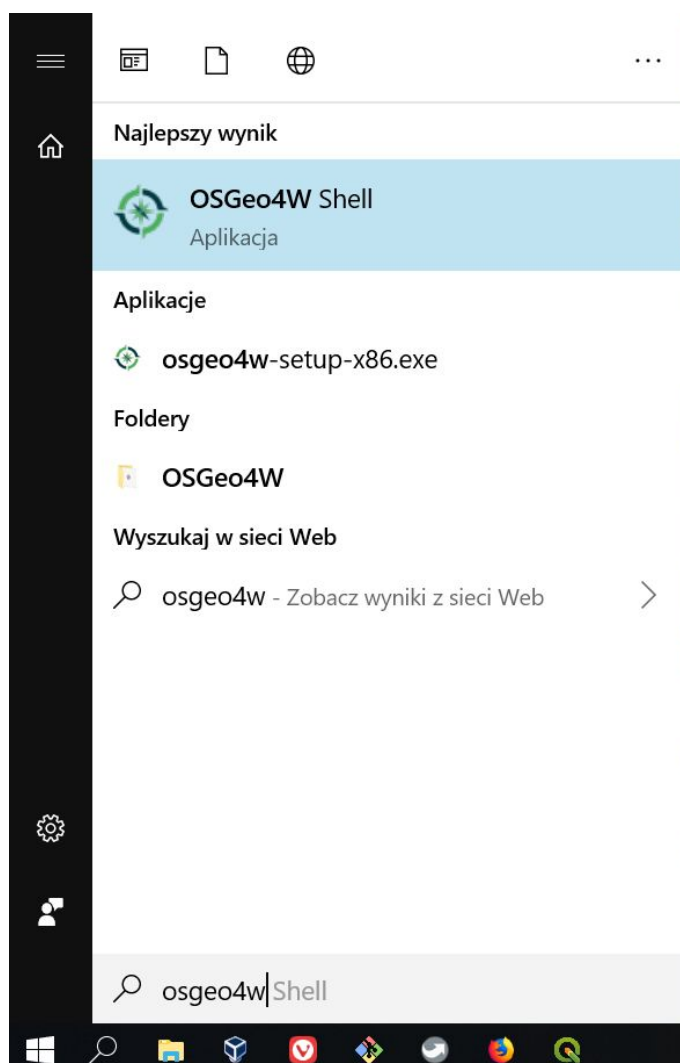
# Ćwiczenie 1. Uruchomienie Pythona 3 w konsoli OSGeo4W Shell

## Treść zadania

Uruchom konsolę *OSGeo4W Shell* i uruchom interpreter Pythona 3.

## Opis

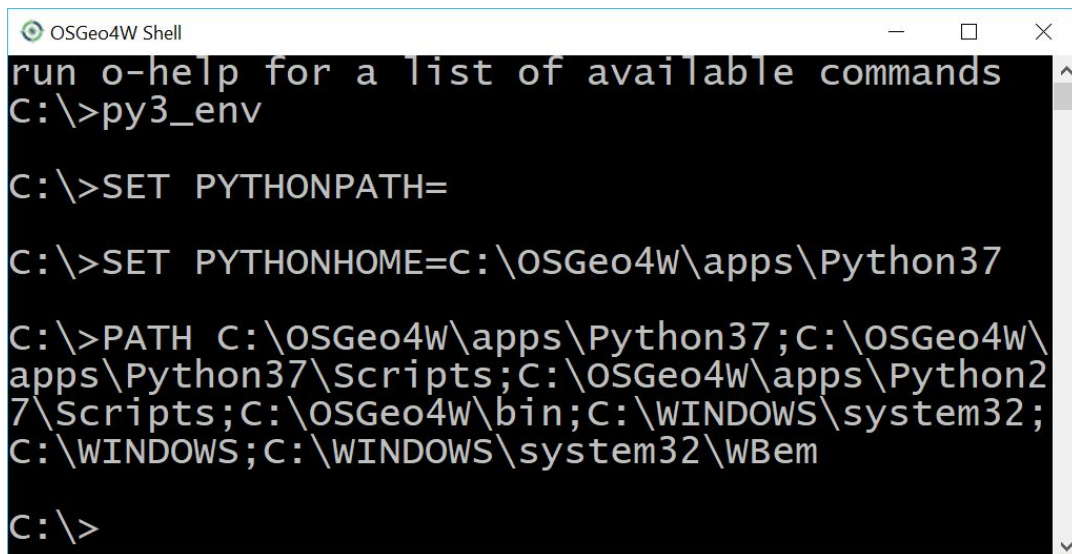
Aby uruchomić konsolę *OSGeo4W Shell* należy wyszukać odpowiednie polecenie w menu Start systemu Windows.



Pojawi się okno konsoli. Domyślnie ustawiony jest w niej Python w wersji 2. Aby przełączyć się na nowszą wersję należy wpisać polecenie `py3_env`. Po jego wpisaniu w konsoli pojawi się kilka wpisów informujących o zmianach w zmiennych środowiskowych.

### GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl



```
OSGeo4W Shell
run o-help for a list of available commands
C:\>py3_env

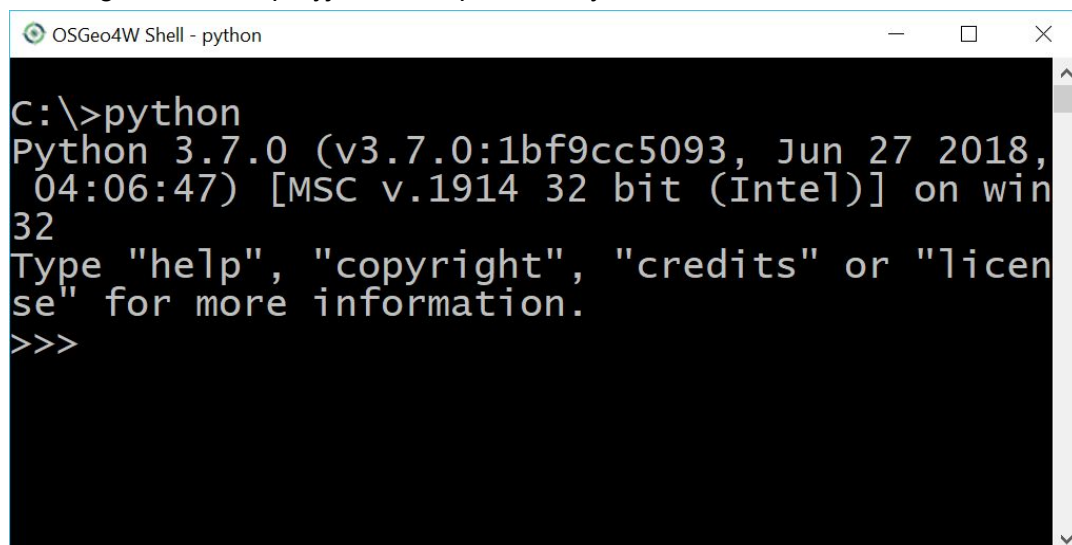
C:\>SET PYTHONPATH=

C:\>SET PYTHONHOME=C:\OSGeo4W\apps\Python37

C:\>PATH C:\OSGeo4W\apps\Python37;C:\OSGeo4W\apps\Python37\Scripts;C:\OSGeo4W\apps\Python27\Scripts;C:\OSGeo4W\bin;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\system32\WBem

C:\>
```

Aby uruchomić interpreter Pythona należy wpisać polecenie `python`. Jeśli wszystko działa poprawnie w konsoli pojawi się informacja o wersji uruchomionego interpretera (powinna być większa niż 3.6), a kursor powinien migać w linii rozpoczynającej się od znacznika `>>>`. Oznacza to gotowość do przyjmowania poleceń Pythona.



```
OSGeo4W Shell - python
C:\>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
```

## Ćwiczenie 2. Praca bezpośrednio w interpreterze Pythona

Po uruchomieniu interpretera ([ćwiczenie 1](#)) możliwe jest wpisywanie poleceń Pythona. Po wciśnięciu klawisza `Enter` wpisane polecenie zostanie wykonane, a wynik działania (jeśli jakiś jest) zostanie wyświetlony w konsoli. Wpisz poniższe polecenia Pythona każde potwierdzając klawiszem `Enter`:

- `x = 2`
- `y = 12`
- `print( x )`
- `print( y )`

Po zakończeniu pracy zamknij interpreter kombinacją klawiszy `Ctrl+Z` i `Enter`.

## Ćwiczenie 3. Stworzenie i uruchomienie skryptu

### Treść

Należy stworzyć skrypt pythona w pliku `skrypt.py` z poniższym kodem źródłowym:

```
#Przypisanie wartości do zmiennych
x = 2
y = 12
#Wydrukowanie wartości zmiennych
print( x )
print( y )
```

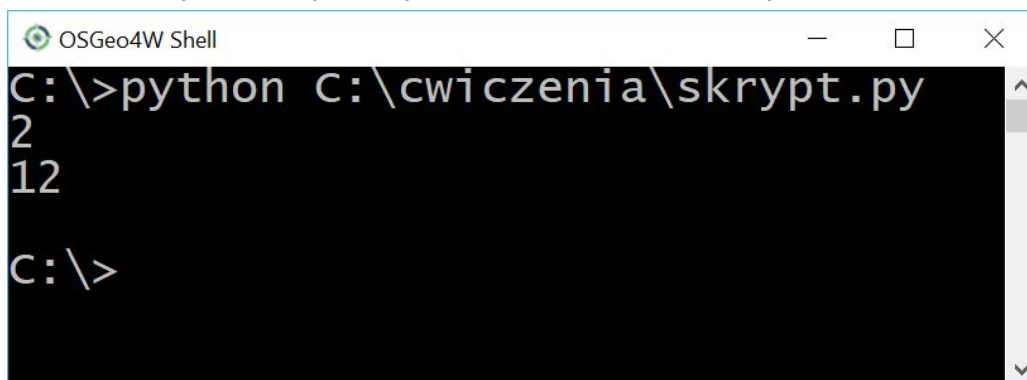
Uruchom powyższy kod za pomocą interpretera Pythona.

### Opis

Aby uruchomić skrypt w interpreterze należy uruchomić konsolę OSGeo4W i ustawić Pythona w wersji 3 (zgodnie z [ćwiczeniem 1](#)). Następnie należy wpisać polecenie:

```
python <katalog>/skrypt.py
```

gdzie za `<katalog>` należy podstawić pełną ścieżkę folderu z plikiem `skrypt.py` np. `C:\cwiczenia`. Wynikiem będzie wyświetlenie wartości zmiennych `x` i `y` w konsoli.



```
OSGeo4W Shell
C:\>python C:\cwiczenia\skrypt.py
2
12
C:\>
```

## Ćwiczenie 4. Import obiektów

### Treść zadania

W jednym katalogu należy utworzyć dwa skrypty Pythona. W pliku `biblioteka.py` należy zdefiniować dwie zmienne `x=10`, `y=5`. W drugim pliku `aplikacja.py` należy zaimportować obie zmienne i wydrukować ich wartości poleceniem `print()`.

## Kod źródłowy

biblioteka.py

```
x = 10  
y = 5
```

aplikacja.py

```
#Import obiektów z biblioteki  
from biblioteka import x, y  
  
print( x, y )
```

## Ćwiczenie 5. Instalacja nowej biblioteki

### Treść zadania

Zainstaluj w Python 3 bibliotekę `primenumbers` do operacji na liczbach pierwszych. Po instalacji sprawdź w interpreterze czy biblioteka została poprawnie zainstalowana wpisując następujące polecenia:

- `import primenumbers`
- `primenumbers.isprime( 3 )` → True
- `primenumbers.isprime( 10 )` → False

### Opis

Aby zainstalować nową bibliotekę należy skorzystać z narzędzia `pip`. Instalacja zostanie dokonana na wersji Pythona, która aktualnie jest aktywna w konsoli, czyli w przypadku wersji 3 należy ją ustawić w `OSGeo4W Shell` poleceniem `py3_env`. Następnie wpisujemy polecenie:

```
python -m pip install primenumbers
```

Następnie należy uruchomić interpreter Pythona i zaimportować moduł `primenumbers`. Jeśli instalacja przebiegła pomyślnie w trakcie importu nie zostanie wywołany żaden błąd.



```
OSGeo4W Shell - python
c:\>python -m pip install primenumbers
Collecting primenumbers
  Downloading https://files.pythonhosted.org/packages/06/f2/54a8bd356e8139e6e0d1dabce9d12f1919ba6ac7c7fd5097cda34e787750/primenumbers-1.1.2.zip
Installing collected packages: primenumbers
  Running setup.py install for primenumbers ... done
Successfully installed primenumbers-1.1.2

You are using pip version 19.0.3, however version 19.2.3 is available.
You should consider upgrading via the 'python -m pip install --upgrade pip' command.

c:\>python
Python 3.7.0 (v3.7.0:1bf9cc5093, Jun 27 2018, 04:06:47) [MSC v.1914 32 bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more in
formation.
>>> import primenumbers
>>> primenumbers.isprime( 3 )
True
>>> primenumbers.isprime( 10 )
False
>>>
```

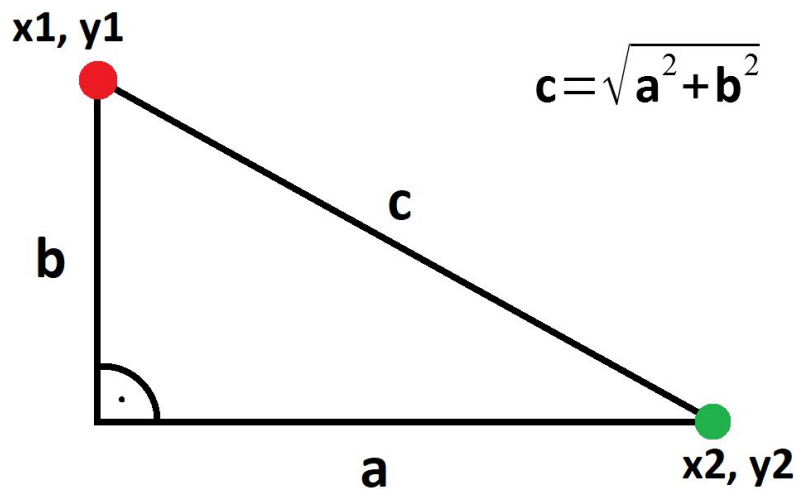
## Ćwiczenie 6. Odległość pomiędzy punktami

### Treść zadania

Należy stworzyć algorytm do obliczenia odległości pomiędzy dwoma punktami w układzie współrzędnych. Współrzędne pierwszego punktu określone są zmiennymi  $x_1$ ,  $y_1$ , a drugiego  $x_2$ ,  $y_2$ . Wynik należy wydrukować poleceniem `print()`.

### Opis algorytmu

Do obliczania odległości pomiędzy dwoma punktami w przestrzeni dwuwymiarowej najprościej jest skorzystać z twierdzenia pitagorasa.



Oba punkty można wpisać w trójkąt prostokątny, znajdując się one w jego wierzchołkach nie mających kąta prostego.

Kroki algorytmu:

1. Obliczenie długości przyprostokątnych  $a$  i  $b$  wykorzystując współrzędne obu punktów.
2. Obliczone wartości podnieść do kwadratu.
3. Zsumować kwadraty długości obu boków.
4. Obliczyć pierwiastek drugiego stopnia z sumy kwadratów. Funkcja `sqrt` do wyliczenia pierwiastka kwadratowego znajduje się w module `math`.

Otrzymany wynik jest długością przeciwprostokątnej, czyli odległością między dwoma punktami.

## Kod źródłowy

```
#Import funkcji do pierwiastkowania
from math import sqrt

#Dane wejściowe algorytmu, współrzędne obu punktów
x1=0
y1=0
x2=1
y2=0

#Obliczanie długości przyprostokątnych
a = x1-x2
b = y1-y2

#Podniesienie wartości do kwadratu
a2 = a**2
```

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

```
b2 = b**2
#Obliczanie długości przeciwprostokątnej - odległości pomiędzy
punktami
c2 = a2+b2
c = sqrt( c2 )
#Wydrukowanie wyniku
print( c )
```

## Ćwiczenie 7. Sprawdzenie czy losowa liczba jest liczbą pierwszą

### Treść zadania

Z pomocą modułu `random` wylosuj liczbę całkowitą z zakresu 1-100. Następnie wydrukuj jej wartość oraz informację czy podana liczba jest liczbą pierwszą. Do sprawdzenia czy liczba jest pierwsza wykorzystaj moduł `primenumbers`.

### Opis

Moduł `random` posiada funkcję `randint`, która przyjmuje dwa argumenty liczbowe określające zasięg losowania. Zwraca ona losową liczbę całkowitą z podanego zasięgu. Do sprawdzenia czy dana liczba jest pierwsza należy wykorzystać funkcję `primenumbers.isprime`.

### Kod źródłowy

```
#Import funkcji z bibliotek
from random import randint
from primenumbers import isprime

#Wylosowanie liczby z podanego zakresu
a = randint(1, 100)

#Warunek sprawdzający Sprawdzenie czy wylosowana liczba
#jest pierwsza
if isprime( a ):
    print( a, '- liczba pierwsza' )
else:
    print( a, '- liczba nie jest pierwsza' )
```

## Ćwiczenie 8. Znajdowanie liczb pierwszych w danym zasięgu

### Treść zadania

Znajdź wszystkie liczby pierwsze w zasięgu liczb naturalnych od 1 do 100. W tym celu należy wykorzystać pętlę `for..in`. Do sprawdzenia czy dana liczba jest pierwsza wykorzystaj moduł `primenumbers`.

Znalezione liczby wydrukuj poleceniem `print()`.

### Opis zadania

Aby znaleźć wszystkie liczby pierwsze w podanym zasięgu należy wykonać pętlę `for` z odpowiednią liczbą iteracji, przy czym każda iteracja będzie dotyczyła kolejnej liczby. W pętli należy stworzyć odpowiednią instrukcję warunkową sprawdzającą czy dana liczba jest pierwsza za pomocą funkcji `primenumbers.isprime`, jeśli warunek będzie spełniony to należy wydrukować daną liczbę.

### Kod źródłowy

```
from primenumbers import isprime

#Iteracja po wszystkich liczbach z podanego zakresu
for n in range(1, 101):
    #Sprawdzenie czy liczba jest pierwsza
    if isprime( n ):
        print( n )
```

## Ćwiczenie 9. Formatowanie tekstu

### Treść zadania

Wykorzystując formatowanie tekstu w Pythonie wydrukuj odpowiednie informacje dla każdej liczby z zakresu od 1 do 10:

- jeśli dana liczba jest pierwsza: *Liczba X jest pierwsza*
- w przeciwnym wypadku: *Liczba X nie jest pierwsza*

### Opis zadania

Wykorzystując pętlę `for` należy wykonać iterację po wskazanym zakresie liczb i wykorzystując instrukcję warunkową `if` sprawdzić czy dana liczba jest pierwsza. W zależności od wyniku warunku należy i wydrukować odpowiedni tekst za pomocą funkcji format łańcuchów znaków.

## Kod źródłowy

```
from primenumbers import isprime

for n in range(1, 11):
    if isprime( n ):
        print( 'Liczba {} jest pierwsza'.format( n ) )
    else:
        print( 'Liczba {} nie jest pierwsza'.format( n ) )
```

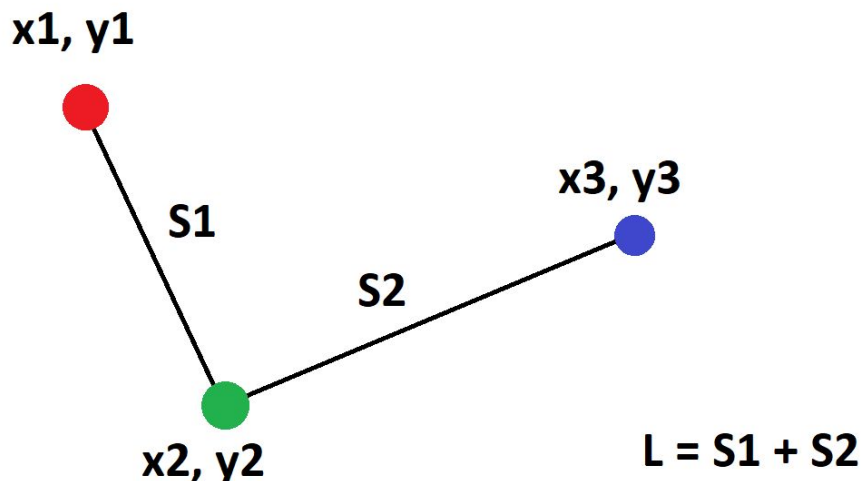
## Ćwiczenie 10. Długość linii

### Treść zadania

Każdy element listy stanowi dwuelementowa tupla definiująca pojedynczy punkt w przestrzeni dwuwymiarowej, gdzie pierwszy element to współrzędna X, a drugi współrzędna Y. W ten sposób otrzymujemy listę wierzchołków opisujących linię np.  $[(0,0), (1,0), (1,1)]$ . Stwórz algorytm do obliczania długości linii zdefiniowanej w ten sposób. Wynik należy wydrukować poleceniem `print()`.

### Opis algorytmu

Każdy element listy stanowi pojedynczy punkt.



Należy obliczyć odległość pomiędzy kolejnymi parami punktów na liście i jako wynik podać sumę tych wartości. Do obliczenia odległości pomiędzy dwoma punktami stosuje się twierdzenie pitagorasa.

Kroki algorytmu:

1. Przed rozpoczęciem pętli należy zdefiniować dwie dodatkowe zmienne do przechowywania długości linii (wartość początkowa 0) i numeru iteracji (wartość początkowa 1).
2. Stworzenie pętli `while`, która wykona się tyle razy ile segmentów ma dana linia, czyli o jeden raz mniej niż jest elementów na liście.
3. Wewnątrz pętli należy wyciągnąć sąsiadującą ze sobą parę punktów wykorzystując zmienną określającą numer iteracji.
4. Wykorzystując kod stworzony w [Ćwiczenie](#) należy obliczyć odległość między daną parą punktów.
5. Wartość zmiennej przechowującej długość linii zwiększyć o obliczoną odległość.

Po zakończeniu pętli długość linii znajduje się w zdefiniowanej na początku zmiennej.

## Kod źródłowy

```
#Import funkcji do pierwiastkowania
from math import sqrt
#Linia z wierzchołkami
linia = [(0,0), (1,0),(1,1)]
#Pomocnicza zmienna przechowująca długość linii
#Przy każdej iteracji będzie zwiększana o długość aktualnego segmentu
wynik = 0
#Pomocnicza zmienna określająca ile iteracji zostało wykonanych
#Służy do wyciągnięcia pary sąsiadujących punktów
iteracja = 1
#Pętla wykonywana dopóki liczba iteracji nie przekroczy liczby segmentów linii
while iteracja<=len(linia)-1:
    #Wyciągnięcie współrzędnych obu punktów
    x1,y1 = linia[iteracja-1]
    x2,y2 = linia[iteracja]
    #Obliczenie długości segmentu
    a = x1-x2
    b = y1-y2
    a2 = a**2
    b2 = b**2
    c2 = a2+b2
    c = sqrt( c2 )
    #Dodanie długości segmentu do końcowego wyniku
    wynik += c
    #Określenie numeru kolejnej iteracji
    iteracja += 1
#Wydrukowanie wyniku
print( wynik )
```

# Ćwiczenie 11. Generowania słownika z liczb

## Treść zadania

Dla liczb od 1 do 9 stwórz słownik, gdzie klucz to dana liczba, a wartość stanowi ta liczba podniesiona do kwadratu.

Wygenerowany słownik wydrukuj poleceniem `print()`.

## Opis

Na początku należy zdefiniować pusty słownik, który będzie wypełniany kolejnymi wartościami. Następnie należy wykonać odpowiednią liczbę iteracji, np. wykorzystując pętlę `for`, po kolejnych liczbach naturalnych z podanego zakresu. Można to zrobić z wykorzystaniem funkcji `range(1, 10)`. W pętli należy obliczyć potęgę drugiego stopnia danej liczby i przypisać jej do klucza, którym jest liczba bazowa.

## Kod źródłowy

```
#Pusty słownik, który będzie wypełniany wartościami
sownik = {}
#Iteracja po kolejnych liczbach naturalnych
for liczba in range(1,10):
    #Obliczenie kwadratu liczby
    potega = liczba**2
    #Przypisanie do odpowiedniego klucza
    sownik[liczba] = potega
print( sownik )
```

# Ćwiczenie 12. Średnia geometryczna elementów listy

## Treść zadania

Oblicz średnią geometryczną liczb z listy:

```
lista = [2, 4, 8, 16, 32]
```

Wynik wydrukuj poleceniem `print()`.

## Opis

Średnia geometryczna dla  $n$  liczb to pierwiastek  $n$ -tego stopnia z iloczynu wszystkich tych liczb. Należy więc wykonać dwa główne kroki:

- obliczyć iloczyn liczb
- wyliczyć pierwiastek  $n$ -tego stopnia

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI  
Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

Do wykonania pierwszego kroku należy wykorzystać pętlę `for` po podanej liście. Przed samą pętlą należy stworzyć pomocniczą zmienną, która będzie przechowywała aktualną wartość iloczynu. Jej początkowa wartość musi wynosić 1, tak, żeby przy pierwszej iteracji, po wykonaniu mnożenia, otrzymać pierwszą liczbę z listy.

Po zakończeniu pętli w pomocniczej zmiennej będzie iloczyn wszystkich liczb z listy. W Python nie ma funkcji do obliczania pierwiastka dowolnego stopnia. Łatwo jednak go wyliczyć podnosząc pierwiastkowaną liczbę do potęgi  $1/n$ . Po wykonaniu tej operacji jej wynik należy wydrukować poleceniem `print()`.

## Kod źródłowy

```
lista = [2, 4, 8, 16, 32]

#Pomocnicza zmienna przechowująca iloczyn kolejnych liczb
iloczyn = 1
for liczba in lista:
    #Mnożenie kolejnych liczb
    iloczyn *= liczba
#Długość listy
n = len( lista )
#Obliczenie średniej geometrycznej
srednia_geometryczna = iloczyn**(1/n)
print( srednia_geometryczna )
```

## Ćwiczenie 13. Totolotek

### Treść zadania

Zdefiniuj listę `skreslone` i wpisz do niej sześć losowych liczb z zakresu od 1 do 19 np. `[10, 15, 2, 13, 4, 9]`. Następnie wykorzystując funkcję `random.randint` wylosuj kolejne sześć liczb i zapisz je do nowej listy `wylosowane`. Wylosowane liczby nie mogą się powtarzać. Porównaj otrzymane liczby ze skreślonymi i zapisz ile z nich się powtarza w obu listach.

W zależności od otrzymanego wyniku wydrukuj jeden z poniższych komunikatów:

- 6 - Główna nagroda
- 3-5 - Nagroda pieniężna za trafienie X liczb
- <3 - Spróbuj ponownie

### Opis

Na początku definiujemy dwie listy, pierwsza 6-elementowa z wybranymi liczbami, druga pusta, do której trafią wylosowane liczby. Następnie należy wylosować 6 liczb i wstawić je do listy `wylosowane`. W związku z tym, że wartości muszą być unikalne należy zastosować pętlę `while`, która będzie się wykonywała dopóki liczba elementów w liście nie osiągnie

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
[www.gis-support.pl](http://www.gis-support.pl), [info@gis-support.pl](mailto:info@gis-support.pl)



zakładanej wartości. Wewnątrz pętli należy zdefiniować odpowiedni warunek sprawdzający czy wylosowana liczba jest w liście, jeśli nie to można ją dodać.

Następnie należy porównać obie listy. Można to zrobić iterując po elementach jednej z nich i sprawdzając czy jest on również w drugiej liście. Każde trafienie zapisujemy w osobnej zmiennej.

Na koniec, mając liczbę trafień, należy stworzyć warunki sprawdzające ile liczb trafiono i w zależności od wyniku wydrukować odpowiedni komunikat. W przypadku trafienia 3, 4 lub 5 liczb dodatkowo należy wykonać formatowanie tekstu aby podać ile liczb trafiono.

## Kod źródłowy

```
from random import randint

#Lista z liczbami wybranymi przez użytkownika
skreslone = [10, 15, 2, 13, 4, 9]
#Pusta lista, w której zapisane zostaną wylosowane liczby
wylosowane = []
#Pętla, która będzie się wykonywała dopóki lista się nie zapełni
while len(wylosowane)<6:
    #Wylosowanie liczby
    liczba = randint(1,20)
    #Dodanie liczby możliwe tylko jeśli nie ma jej w liście
    if not (liczba in wylosowane):
        wylosowane.append( liczba )
#Pomocnicza zmienna określająca ile liczb powtarza się w obu listach
trafione = 0
#Iterujemy po jednej z list
for liczba in skreslone:
    #Jeśli liczba jest w drugiej liście to zwiększamy wartość zmiennej
    if liczba in wylosowane:
        trafione += 1
#W zależności od wyniku drukujemy odpowiedni komunikat
if trafione == 6:
    print( 'Główna nagroda!' )
elif trafione > 2:
    print( 'Nagroda pieniężna za trafienie {} liczb'.format( trafione ) )
else:
    print( 'Spróbuj ponownie' )
```

# Ćwiczenie 14. Funkcja do obliczania obwodu i pola koła

## Treść zadania

Stwórz funkcję `kolo` do obliczania obwodu i pola koła. Jako argument przyjmuje ona wartość promienia. Liczba `pi` powinna zostać zaimportowana z modułu `math`. Jako wynik działania należy zwrócić słownik zawierający dwa elementy o kluczach `obwód` i `pole` przechowujące obliczone wartości.

Jako wynik wydrukuj otrzymany słownik dla promienia 5.

## Opis

Na początku, zgodnie z poleceniem, należy zaimportować wartość liczby `pi` z modułu `math`. Następnie należy stworzyć funkcję, która przyjmuje argument określający promień koła. Wewnątrz funkcji kod musi być odpowiednio wcięty o jeden poziom w prawo. Następnie można wykonać obliczenia obwodu i pola koła na podstawie podanego promienia. Aby zwrócić wynik należy stworzyć słownik z kluczami `obwód` i `pole` i przypisać im obliczone wartości.

Po zakończeniu funkcji wracamy na główny poziom skryptu i wywołujemy stworzoną funkcję podając jako argument wartość liczbową. Wynik działania należy wydrukować poleceniem `print()`.

## Kod źródłowy

```
from math import pi

#Definicja funkcji przyjmującej pojedynczy argument
def kolo(promien):
    #Obliczenie obwodu koła
    obwod = 2 * pi * promien
    #Obliczenie pola koła
    pole = pi * (promien**2)
    #Stworzenie słownika z wynikami
    #i zwrócenie jako wynik działania funkcji
    return {'obwód':obwod, 'pole':pole}

#Wywołanie funkcji
wynik = kolo(5)
print( wynik )
```

# Ćwiczenie 15. Funkcje do obliczania odległości i długości

## Treść zadania

Stwórz dwie funkcje:

- `odleglosc( punkt1, punkt2 )` - przyjmuje dwa argumenty, oba są dwuelementowymi tuplami definiującymi dwa punkty, jak wynik zwraca odległość między tymi punktami,
- `dlugosc( linia )` - przyjmuje jeden argument, linię opisaną listą zawierającą jako dwuelementowe tuple opisujące jej wierzchołki, zwraca długość tej linii, np. `[(0,0), (1,0), (1,1)]`

Wynik działania funkcji `dlugosc` należy wydrukować poleceniem `print()`.

## Opis

Funkcja `odleglosc` ma dwa argumenty reprezentujące punkty. Jako wynik należy podać odległość między nimi obliczoną na podstawie twierdzenia pitagorasa.

Funkcja `dlugosc` przyjmuje listę punktów w formie dwuelementowych tupli. Należy wykonać iterację po sąsiadujących parach punktów i dla każdej pary obliczyć odległość między nimi. Jako wynik należy podać sumę obliczonych wartości.

## Kod źródłowy

```
#Import funkcji do pierwiastkowania
from math import sqrt

#Funkcja obliczająca odległość pomiędzy podanymi dwoma punktami
#Oba punkty są określone jako dwuelementowe tuple ze współrzędnymi XY
def odleglosc( punkt1, punkt2 ):
    #Rozpakowanie współrzędnych do zmiennych
    x1,y1 = punkt1
    x2,y2 = punkt2
    #Obliczenie długości segmentu
    a = x1-x2
    b = y1-y2
    a2 = a**2
    b2 = b**2
    c2 = a2+b2
    return sqrt( c2 )

#Funkcja do obliczania długości linii określonej listą zawierającą dwuelementowe tuple
def dlugosc( linia ):
```

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

```
wynik = 0
iteracja = 1
while iteracja<=len(linia)-1:
    #Wyciągnięcie sąsiednich punktów
    punkt1 = linia[iteracja-1]
    punkt2 = linia[iteracja]
    #Obliczenie odległości między punktami
    wynik += odleglosc( punkt1, punkt2 )
    #Określenie numeru kolejnej iteracji
    iteracja += 1
return wynik

#Linia z wierzchołkami
wynik = dlugosc( [(0,0), (1,0),(1,1)] )
#Wydrukowanie wyniku
print( wynik )
```

## Ćwiczenie 16. Klasy punktów i linii

### Treść zadania

Stwórz dwie klasy:

- `Punkt` - opisuje punkt dwuwymiarowy, zawiera dwa atrybuty `x` i `y`, których wartości są ustalane w momencie tworzenia instancji klasy oraz metodę `odleglosc`, która przyjmuje jako argument inny punkt i zwróci odległość między nimi,
- `Linia` - opisuje linię, zawiera atrybut `wierzcholki` zawierający listę punktów tworzących tą linię oraz metodę `dlugosc`, która zwraca długość linii. Każdy wierzchołek linii jest instancją klasy `Punkt`.

Stwórz instancję klasy `Linia` z wierzchołkami `[Punkt(0,0), Punkt(1,0), Punkt(1,1)]` i wydrukuj wartość zwracaną przez metodę `dlugosc`.

### Opis algorytmu

Obie klasy muszą posiadać zdefiniowaną metodę `__init__`. Klasa `Punkt` ma przyjmować współrzędne `x` i `y` jako argumenty tej metody, klasa `Linia` ma pojedynczy argument - listę punktów tworzących linię. Wewnątrz tej metody należy zapamiętać przekazane parametry aby możliwe było ich użycie w innych miejscach klasy.

W klasie `Punkt` dodajemy nową metodę `odleglosc`, jako jej argument podawany będzie inny punkt (druga instancja klasy `Punkt`). Metoda ma zwracać odległość od punktu określonego daną instancją do punktu podanego jako jej argument.

Na zakończenie należy utworzyć instancję klasy `Linia`, podając jako argument listę punktów (instancje klasy `Punkt` o różnych współrzędnych XY) i wywołać metodę `dlugosc`.

## Kod źródłowy

```
#Import funkcji do pierwiastkowania
from math import sqrt

#Klasa opisująca punkt
class Punkt:

    def __init__(self, x, y):
        #Przy tworzeniu instancji zapamiętujemy wartości podane przez użytkownika
        self.x = x
        self.y = y

    def odleglosc( self, punkt ):
        #Pierwszy punkt określa zmienna 'self', drugi 'punkt'
        #Do obliczeń wykorzystujemy atrybuty obu punktów
        a = self.x - punkt.x
        b = self.y - punkt.y
        a2 = a**2
        b2 = b**2
        c2 = a2+b2
        return sqrt( c2 )

#Klasa linii
class Linia:

    def __init__(self, wierzcholki):
        #Zapamiętanie wartości podanego argumentu
        self.wierzcholki = wierzcholki

    def dlugosc( self ):
        #Dla wygody wyciągamy listę do zmiennej lokalnej
        linia = self.wierzcholki
        wynik = 0
        iteracja = 1
        while iteracja<=len(linia)-1:
            punkt1 = linia[iteracja-1]
            punkt2 = linia[iteracja]
            #Przy obliczaniu odległości między punktami wykorzystujemy
            #metodę 'odleglosc' jednego z punktów,
```

```

        #drugi punkt podajemy jako argument tej metody
        wynik += punkt1.odleglosc( punkt2 )
        iteracja += 1
    return wynik

linia = Linia( [Punkt(0,0), Punkt(1,0), Punkt(1,1)] )
#Wydrukowanie wyniku
print( linia.dlugosc() )

```

## Ćwiczenie 17. Dziedziczenie klas

### Treść zadania

Stwórz klasę `Okrag`, w trakcie tworzenia instancji tej klasy jako argument podawany jest promień. Klasa ma dodatkową metodę `obwod`, która zwraca obwód okręgu. Następnie stwórz klasę `Kolo`, która dziedziczy klasę `Okrag`. Poza metodą `obwod`, klasa ta posiada metodę `pole` zwracającą pole powierzchni danego koła. Wartość `pi` należy pobrać z modułu `math`.

Stwórz instancję klasy `Kolo` o promieniu 10 i wydrukuj poleceniem `print()` jego obwód i powierzchnię.

### Opis

Wartości liczby  $\pi$  należy zaimportować z modułu `math`.

Klasa `Okrag` musi mieć dwie metody:

- `__init__(self, promien)` - metoda wywoływana w momencie tworzenia instancji klasy, musi przyjmować jeden dodatkowy argument określający promień okręgu, którego wartość należy zapamiętać,
- `obwod(self)` - metoda zwracająca obwód okręgu.

Obwód jest liczony z wzoru  $2\pi r$ , promień jest pobierany z atrybutu klasy, wartość  $\pi$  jest określona zmienną `pi` z modułu `math`.

Definiując klasę `Kolo` należy podać klasę `Okrag` w definicji, aby wykonać dziedziczenie. Wszystkie atrybuty i metody zdefiniowane w klasie `Okrag` będą dostępne również w klasie `Kolo`. Dodatkowo należy dopisać metodę `pole`, w której z wzoru  $\pi r^2$  zostanie obliczone i zwrócone pole koła.

Po zdefiniowaniu obu klas należy utworzyć nową instancję klasy `Kolo` o promieniu 10 i wywołać obie metody: `obwod` i `pole`.

### Kod źródłowy

```

from math import pi

#Stworzenie klasy bazowej
class Okrag:

```

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI  
Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

```

def __init__(self, promien):
    #Metoda wywoływana jest w momencie tworzenia instancji klasy
    #Wartość argumentu jest zapamiętywana, dzięki czemu
    #może być wykorzystana w innych metodach klasy
    self.r = promien

def obwod(self):
    #Obliczenie obwodu okręgu
    return 2 * pi * self.r
#Stworzenie instancji klasy potomnej, w nawiasie podana klasa bazowa
class Kolo( Okrag ):
    def pole(self):
        #Metoda zwracająca pole koła
        return pi * (self.r**2)

#Stworzenie instancji klasy, w nawiasie podano wartość argumentu
k = Kolo(10)
#Wywołanie metody zdefiniowanej w klasie potomnej
print( k.pole() )
#Wywołanie metody zdefiniowanej w klasie bazowej
print( k.obwod() )

```

## Ćwiczenie 18. Zapisywanie danych do plików

### Treść zadania

Dany jest słownik:

```
sownik = { 'id': 10, 'opis': 'obiekt punktowy', 'x': 5, 'y': 4.5 }
```

Zapisz go do pliku dane.txt w ten sposób, że każdy element słownika stanowi osobny wiersz pliku. Każdy wiersz natomiast ma postać klucz=wartość.

### Opis

Dane będą zapisywane do pliku wyjściowego więc nowy plik należy otworzyć w trybie do zapisu 'w'.

```
with open( 'plik.txt', 'w' ) as plik:
```

Następnie należy wykonać iterację po elementach słownika. W pliku należy zapisać zarówno klucz jak i wartość więc iteracja powinna wykorzystywać metodę items():

```
for klucz, wartosc in sownik.items():
```

Pojedynczy wiersz musi stanowić tekst w formie klucz=wartość, tak więc należy sformatować odpowiedni łańcuch znaków z wykorzystaniem zmiennych z pętli. Wiersz musi kończyć się znakiem nowej linii aby przy kolejnej iteracji nowy tekst był w osobnej linii:

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

```
wiersz = '{}={}\n'.format( klucz, wartosc )
```

Na koniec należy zapisać tekst do pliku metodą `write()`.

## Kod źródłowy

```
sloownik = { 'id': 10, 'opis': 'obiekt punktowy', 'x': 5, 'y': 4.5 }  
#Otworzenie pliku w trybie do zapisu  
with open( r'C:\Users\ppociask\Desktop\pli.txt', 'w' ) as plik:  
    #Iteracja po elementach słownika  
    for klucz, wartosc in sloownik.items():  
        #Tekst zawierający klucz i wartość elementu słownika  
        wiersz = '{}={}\n'.format( klucz, wartosc )  
        #Zapis danych do pliku  
        plik.write( wiersz )
```



# Ćwiczenia do samodzielnego wykonania

## Ćwiczenie 19. Średnia arytmetyczna

### Treść zadania

Oblicz średnią arytmetyczną liczb z podanej listy:

```
lista = [2, 4, 16, 32]
```

### Wynik

13.5

## Ćwiczenie 20. Modyfikacja tekstu

### Treść zadania

Dany jest ciąg znaków:

```
jeden,dwa,trzy,cztery
```

Napisz kod, który przekształci go zamieniając separator z przecinka na średnik oraz wszystkie wyrazy rozpocznie od dużej litery. Wynik wydrukuj poleceniem `print()`.

### Wynik

```
Jeden;Dwa;Trzy;Cztery
```

## Ćwiczenie 21. Wyszukiwanie elementów list

### Treść zadania

Dana jest lista z cyframi:

```
lista = [2, 7, 3, 5, 1, 6]
```

Stwórz kod, w którym tworzona jest nowa lista zawierająca wszystkie cyfry, których nie ma w podanej liście. Wynik wydrukuj poleceniem `print()`.

### Wynik

```
[0, 4, 8, 9]
```

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI  
Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
[www.gis-support.pl](http://www.gis-support.pl), [info@gis-support.pl](mailto:info@gis-support.pl)

## Ćwiczenie 22. Usuwanie duplikatów z kolekcji

### Treść zadania

Dana jest lista:

```
lista = ['gruszka', 'jabłko', 'śliwka', 'jabłko', 'jabłko',  
'truskawka', 'śliwka', 'jabłko', 'truskawka']
```

Stwórz kod generujący słownik, którego kluczem będzie konkretny element listy, a wartością liczba określająca ile razy dany element występuje w tej liście.

### Wynik

```
{'gruszka':1, 'jabłko': 4, 'śliwka': 2, 'truskawka': 2 }
```

## Ćwiczenie 23. Tekst na słownik

### Treść zadania

Stwórz słownik z podanego łańcucha znaków:

```
tekst = "login=user;password=pass;port=5432;database=name"
```

Poszczególne elementy słownika są rozdzielone średnikiem, a każdy z wydzielonych tekstów ma format `klucz=wartość`.

### Wynik

```
{  
    'login' : 'user',  
    'password' : 'pass',  
    'port' : '5432',  
    'database' : 'name'  
}
```

## Ćwiczenie 24. Przeliczanie stopni

### Treść zadania

Stwórz dwie funkcje: `stopnie` i `radiany`. Każda z nich powinna przyjąć jeden argument liczbowy określający miarę kąta odpowiednio w radianach i stopniach. Wynikiem będzie przeliczenie podanej wartości na drugi typ zapisu kątów, czyli funkcja `stopnie` przyjmując wartość w radianach zwróci kąt w stopniach natomiast funkcja `radiany` wykona odwrotną operację.

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
[www.gis-support.pl](http://www.gis-support.pl), [info@gis-support.pl](mailto:info@gis-support.pl)

## Wynik

```
print( stopnie( 3.1415 ) )  
→ 180  
print( radians( 90 ) )  
→ 1.57...
```

## Ćwiczenie 25. Wyszukiwanie dzielników liczby

### Treść zadania

Stwórz funkcję `dzielniki`, która przyjmie liczbę naturalną i zwróci wszystkie jej dzielniki w formie listy.

### Wynik

```
dzielniki( 10 )  
→ [1, 2, 5]  
dzielniki( 100 )  
→ [1, 2, 4, 5, 10, 20, 25, 50]  
dzielniki( 53 )  
→ [1]
```

## Ćwiczenie 26. Ciąg Fibonacciego

### Treść zadania

Stwórz funkcję `fibonacci`, która przyjmuje jako argument wartość liczbową, a zwróci listę zawierającą kolejne elementy ciągu Fibonacciego. Ilość elementów tej listy musi być równa podanej wartości liczbowej. Jeśli podano liczbę mniejszą od 1 lista powinna być pusta. Ciąg Fibonacciego to ciąg liczb naturalnych, których każdy kolejny element jest sumą dwóch poprzednich: 1, 1, 2, 3, 5, 8, 13, itd.

### Wynik

```
fibonacci( 10 )  
→ [1, 1, 2, 3, 5, 8, 13, 21, 34, 55]  
fibonacci( 1 )  
→ [1]  
fibonacci( -5 )  
→ []
```

## Ćwiczenie 27. Parser plików CSV

GIS Support Sp. z o.o.

Konrada Wallenroda 2f/3.09, 20-607 Lublin, tel. 81 451-14-90, NIP: 9462641761, REGON: 061483531, KRS: 0000440891, VI  
Wydział Gospodarczy KRS Sąd Rejonowy Lublin Wschód z siedzibą w Świdniku, Kapitał zakładowy 5 000 zł  
www.gis-support.pl, info@gis-support.pl

## Treść zadania

Dany jest plik tekstowy `linia.csv` o zawartości:

```
10,5.5
12.3,4
12,6
10,6.5
```

Każdy wiersz tego pliku definiuje pojedynczy punkt, którego współrzędne X i Y są rozdzielone przecinkiem. Napisz skrypt, który otworzy podany plik, odczyta jego zawartość i obliczy długość linii utworzonej przez wszystkie punkty. Kolejność punktów tworzących linię jest zdefiniowana kolejnością wierszy w pliku, pierwszy wiersz to pierwszy wierzchołek linii.

## Wynik

```
6.8298336979736955
```

## Ćwiczenie 28. Zapis listy obiektów do pliku

### Treść zadania

Dana jest lista zawierająca słowniki. Każdy słownik ma ten sam zestaw kluczy:

```
lista = [
    { "x": 0, "y":0.5, "opis": "punkt 1" },
    { "x": 1.3, "y":2, "opis": "punkt 2" },
    { "x": 1.1, "y":1.5, "opis": "punkt 3" },
]
```

Zapisz listę do pliku `punkty.csv`. W pierwszym wierszu pliku znajduje się nagłówek zawierający nazwy kluczy, kolejne wiersze to wartości danych kluczy z kolejnych słowników. Separatorem poszczególnych wartości ma być przecinek.

### Wynik

Plik wynikowy powinien wyglądać w poniższy sposób. Kolejność kolumn może być inna, ale wartości muszą się zgadzać z nagłówkiem tzn. w kolumnie `x` mogą być tylko wartości dla tego klucza.

```
x,y,opis
0,0.5,punkt1
1.3,2,punkt 2
1.1,1.5,punkt 3
```