python

popularny, interpretowany język programowania ogólnego przeznaczenia

twórcą języka jest holenderski programista Guido van Rossum

Python

dynamicznie typowany

używany m.in. do tworzenia:

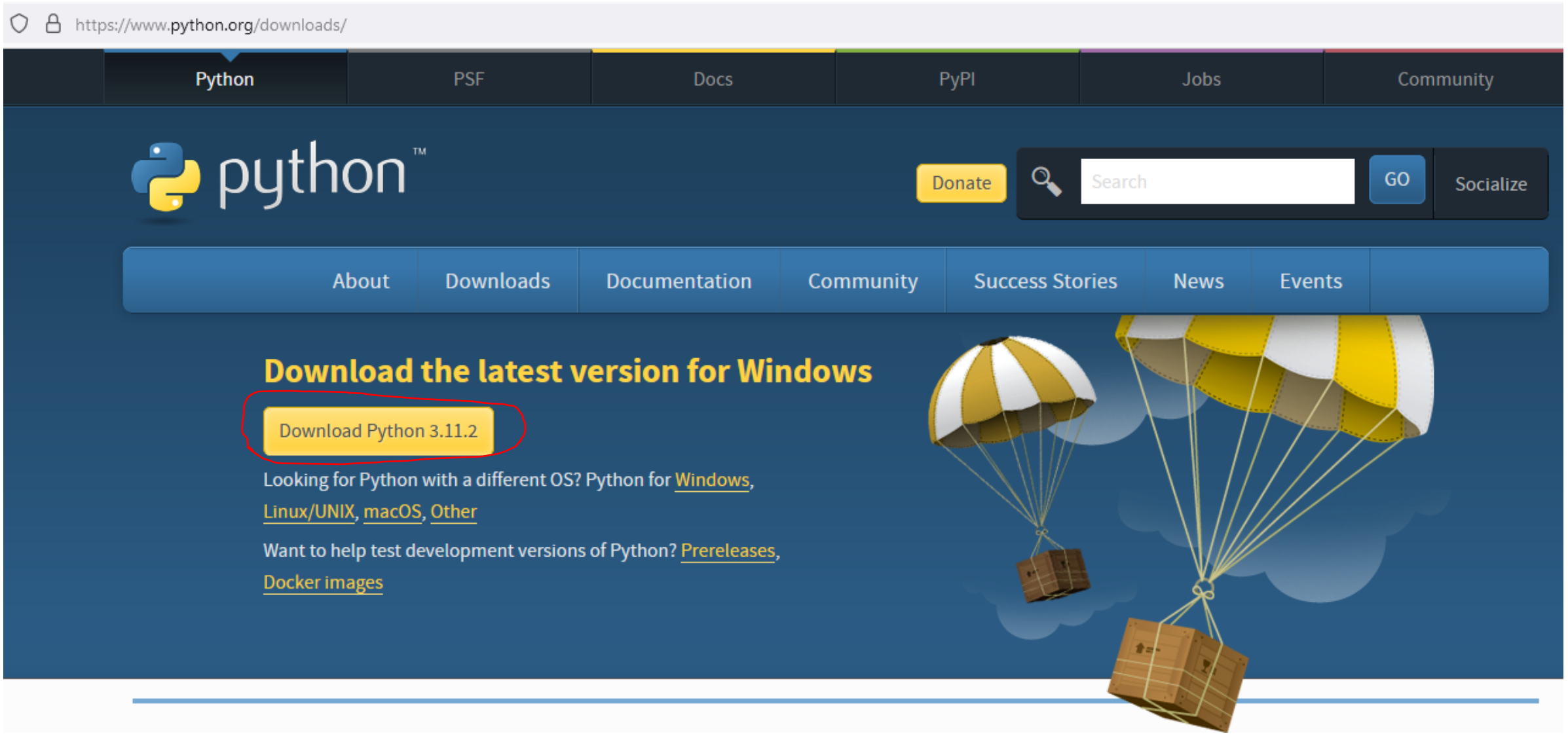
- stron internetowych (po stronie serwera),
- programów desktopowych i mobilnych,
- skryptów systemowych.

główną wersją języka jest Python 3

kładzie nacisk na czytelność kodu przy użyciu wcięć

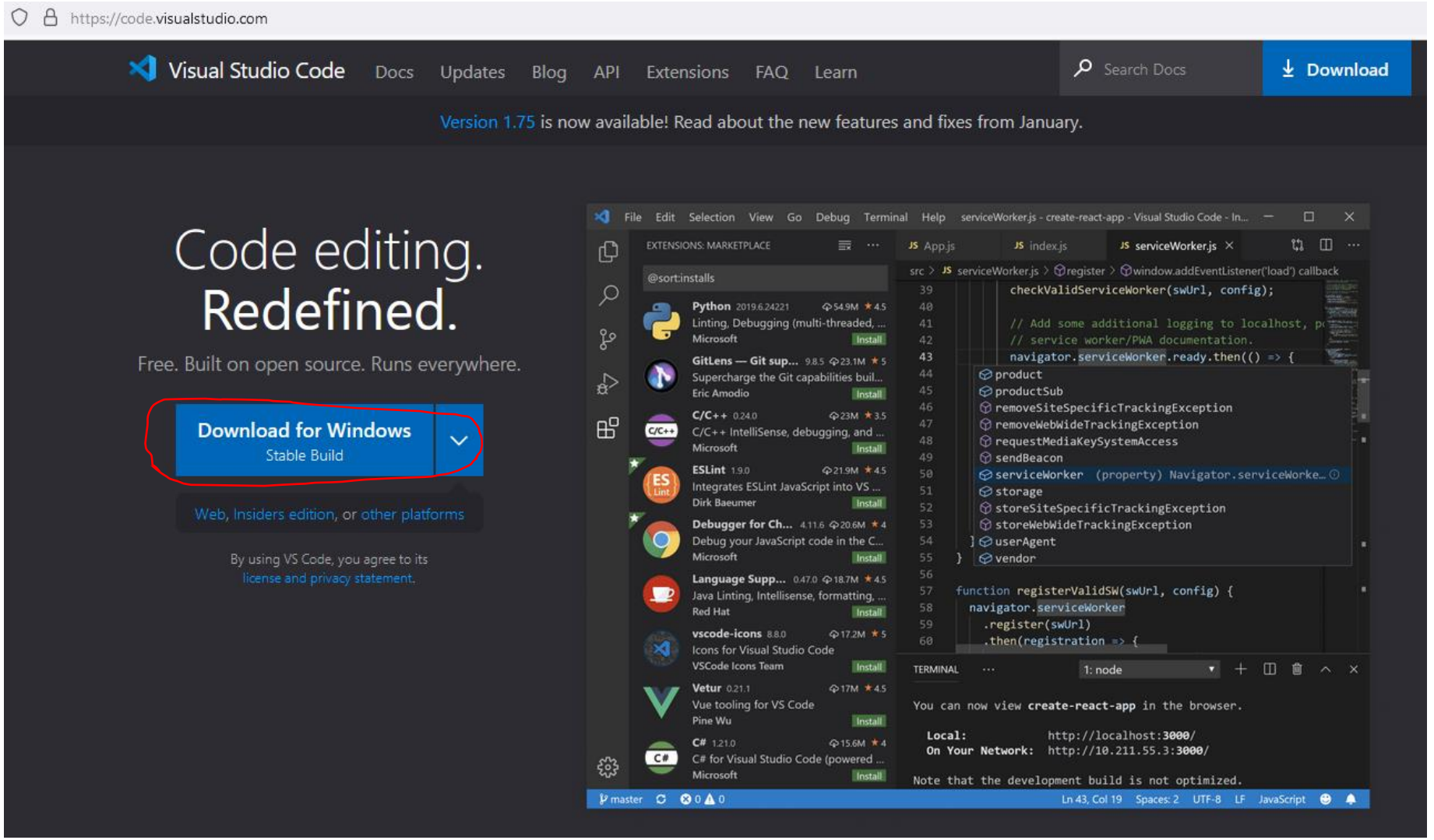
został zaprojektowany z myślą o czytelności i ma pewne podobieństwa do języka angielskiego z wpływem matematyki.

Python download



https://www.python.org/downloads/

Visual Studio Code

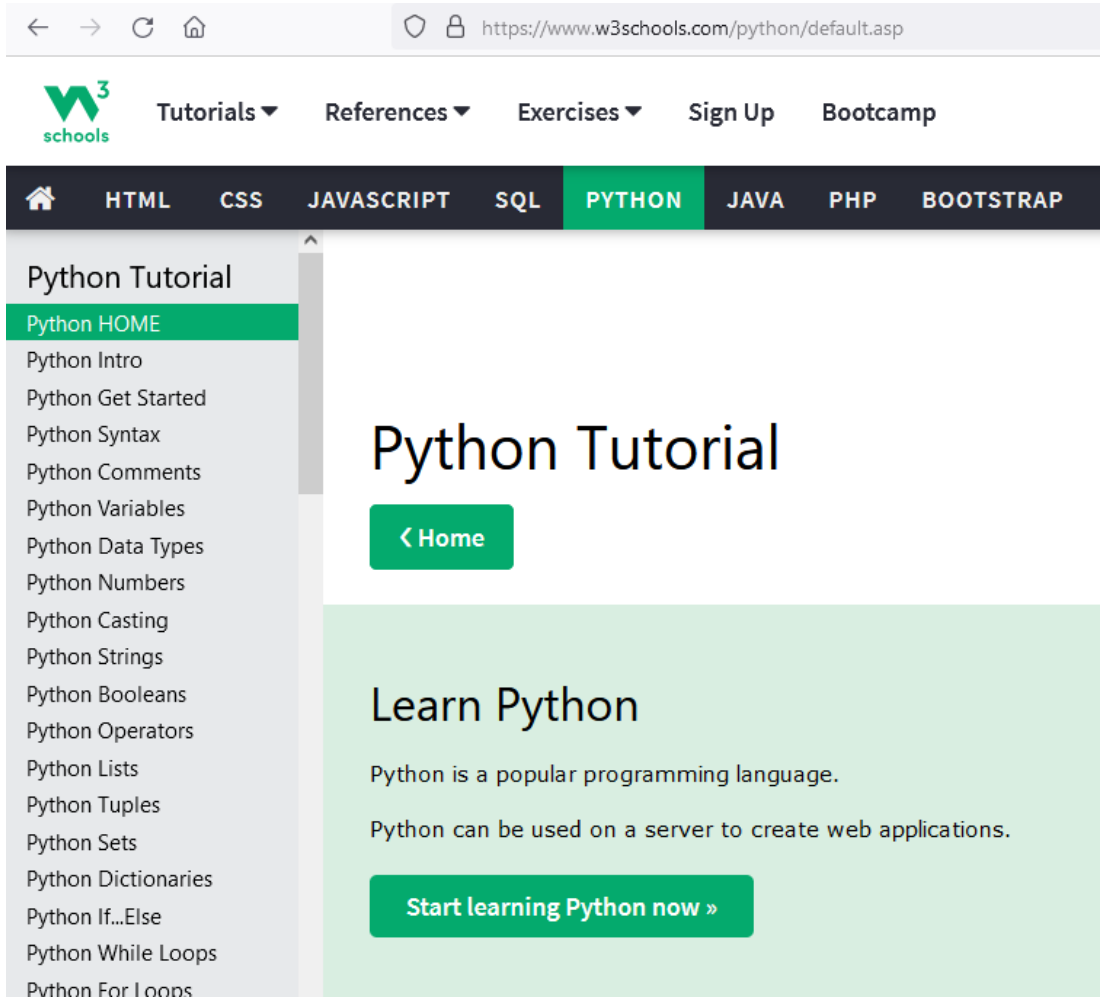


https://code.visualstudio.com/

Python extension for Visual Studio Code

https://code.visualstudio.com/docs/python/python-tutorial

tutorial

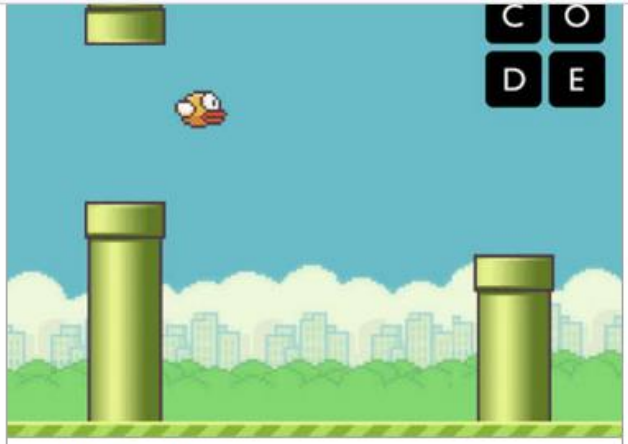


https://www.w3schools.com/python/default.asp

godzina kodowania

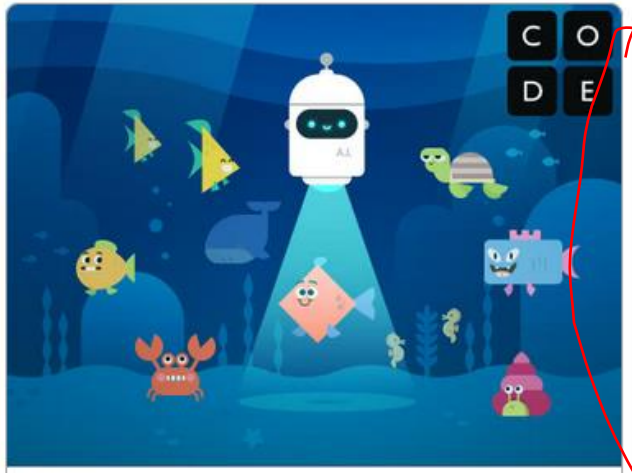
https://hourofcode.com/pl/learn

Minecraft Timecraft
Klasy 2+ | Bloki, Python



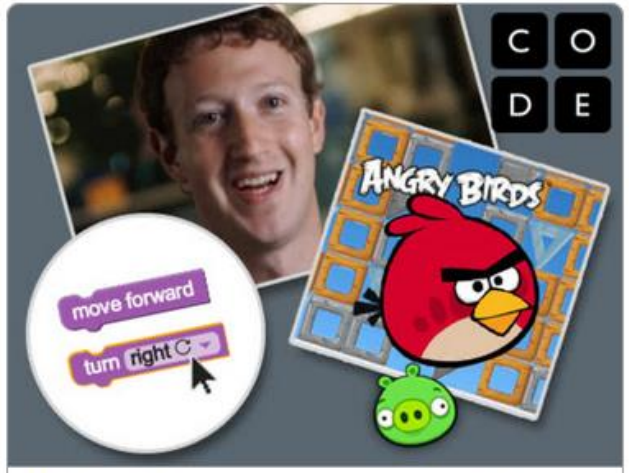
Stwórz grę Flappy
Klasy 2+ | Bloki

Gobliny i Chwała
Klasy 6-8 | JavaScript, Python



AI dla oceanów
Klasy 3+ | AI [sztuczna inteligencja] i nauka maszynowa

Odkryj Python z Compute it
Klasy 2+ | Python | Wszystkie współczesne przeglądarki



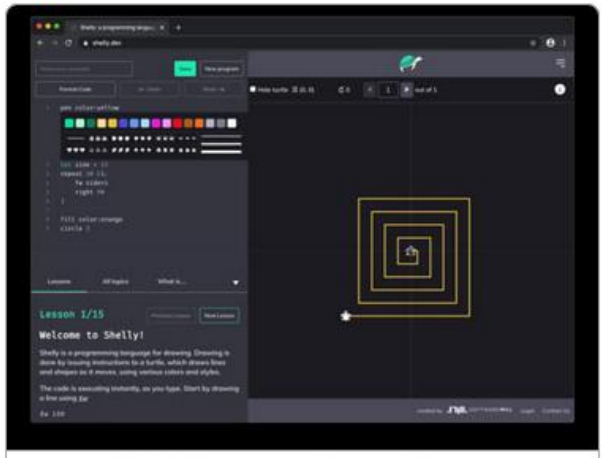
Napisz swój pierwszy program komputerowy
Klasy 2+ | Bloki

https://hourofcode.com/pl/learn

godzina kodowania

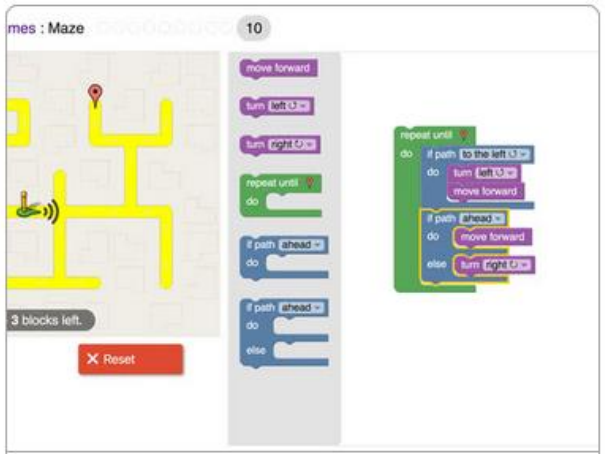
https://hourofcode.com/pl/learn

Anything is Possible! Storytelling with Scratch
Klasy 2-8 | Bloki



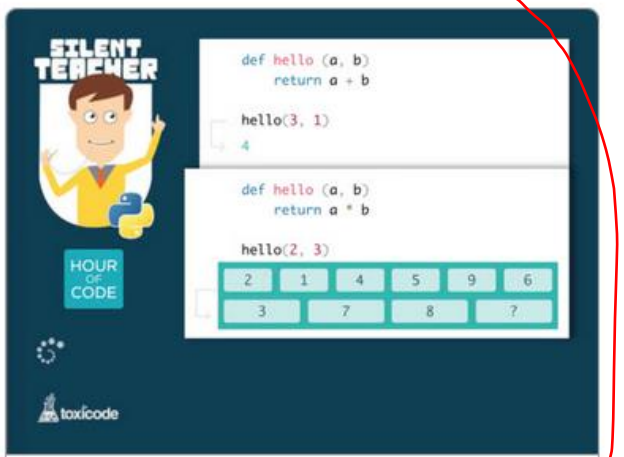
Shelly: Learn programming by drawing!
Klasy 2-8 | A Logo-like language

Kosmiczna rozmowa: Wystrzel w kodzenie w ...
Klasy 2-8 | Bloki, Scratch



Gry w Blockly
Klasy 6+ (od 11 lat) | Bloki

Galaxy Game Jam
Klasy 6+ (od 11 lat) | Bloki | Android



Odkryj Pythona z Silent Teacher
Klasy 6+ (od 11 lat) | Python

https://hourofcode.com/pl/learn

klocki

zmienne
(kontenery do przechowywania danych)

```
a = 5
imie = "Ania"
```

instrukcja warunkowa

```
a = 40
b = 30
if a > b:
  print("a jest większe")
elif a < b:
  print("b jest większe")
else:
  print("a jest równe b")
```

pętla while

```
i = 1
while i < 5:
  print(i)
  i += 1
```

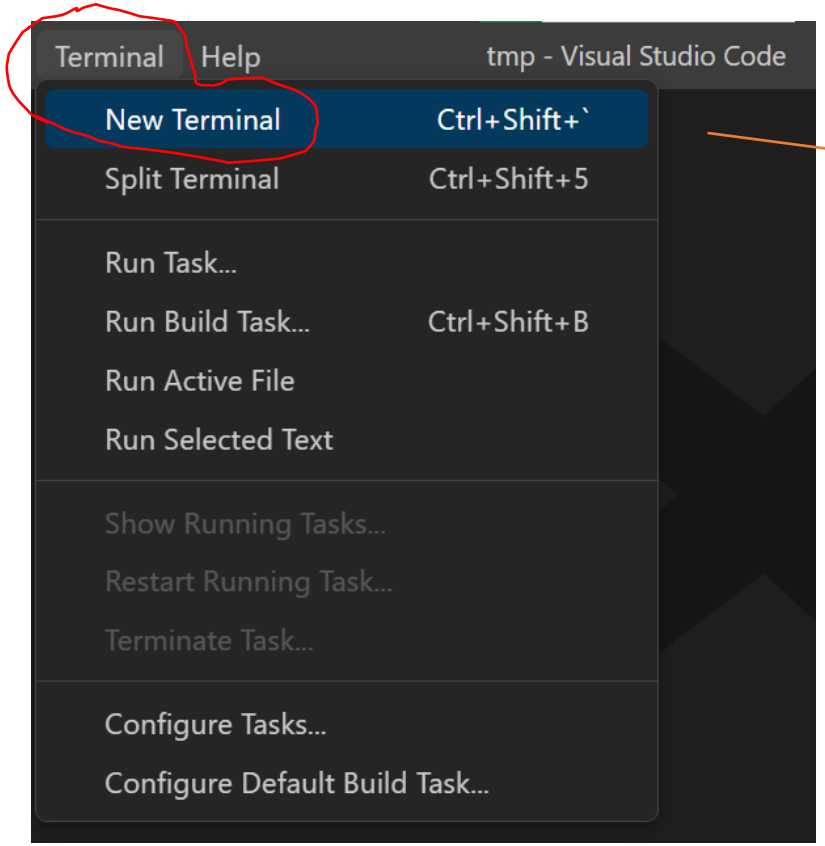
pętla for

```
for x in range(5):
  print(x)
```

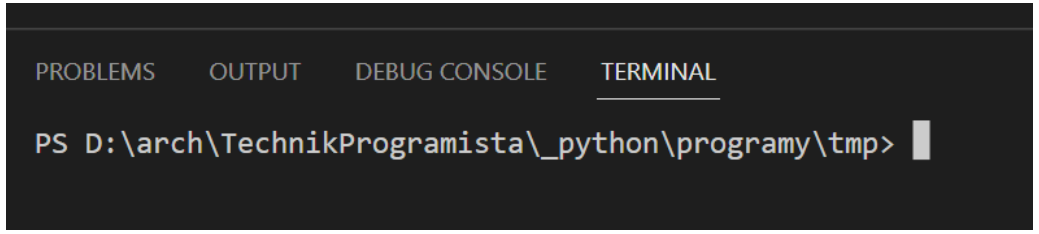
funkcja

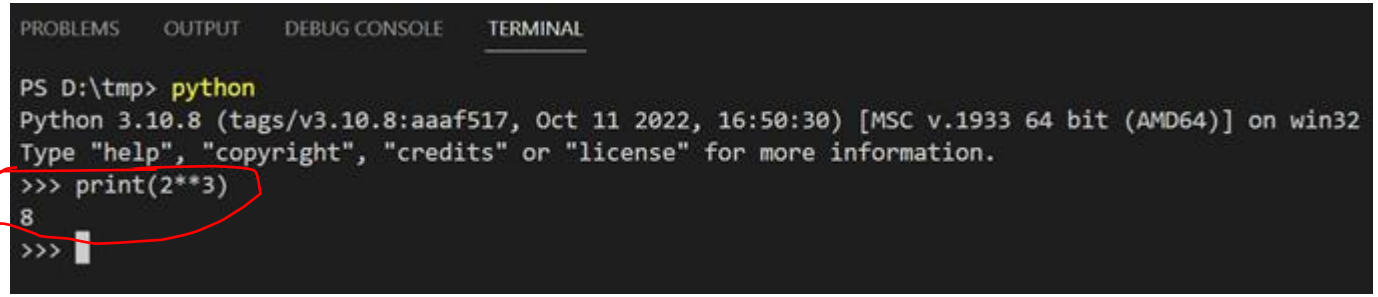
```
def fun():
  print("Hello World")
```

praca w terminalu



Visual Studio Code





zmienne

zmienne są pojemnikami na dane,
które używamy w nszym programie

operator przypisania

nie podajemy nazwy typu
przy nazwie zmiennej -
typowanie dynamiczne
(w zależności od przypisanej wartości interpreter określa typ zmiennej)

liczba = 1

wielkość liter ma znaczenie
(Liczba to nie to samo co liczba)

definicja zmiennej o nazwie *liczba*
i przypisanie jej wartości *1*

```
>>> liczba = 1
>>> liczba
1
>>> print(type(liczba))
<class 'int'>
```

w trakcie działania programu
można zmieniać typ zmiennej

typ zmiennej *int - liczba całkowita*

zmienne – typy danych

```
PS D:\tmp> python
Python 3.10.8 (tags/v3.10.8:aaaf517, Oct 11 2022, 16:50:30) [MSC v.1933 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a = 4
>>> type(a)
<class 'int'>
>>>
>>> b=3.14
>>> type(b)
<class 'float'>
>>>
>>> c="Marian"
>>> type(c)
<class 'str'>
>>>
>>> type(B)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'B' is not defined. Did you mean: 'b'?
```

zmienna typu *int – liczba całkowita*

Python nie ma ograniczenia dla liczb całkowitych (jak to się dzieje w innych językach), które mogą być tak duże, jak duża jest pamięć, która je przechowuje.

zmienna typu *float – liczba rzeczywista*

zmienna typu *string – ciąg znaków*

ciąg znaków może być zamknięty w cudzysłowach lub apostrofach

```
>>> d=True
>>> type(d)
<class 'bool'>
```

zmienna *logiczna – True lub False*

wielkość liter ma znaczenie

https://www.w3schools.com/python/python_datatypes.asp

string

w potrójnym cudzysłowie możemy umieścić ciąg tekstowy w kilku linijkach

```
>>> komp = """
... komputer
... jest
... mocny
... """
>>> komp
'\nkomputer\njest\nmocny\n'
```

List (lista)

uporządkowana kolekcja danych różnych typów

lista może zawierać różne typy danych (również inne listy)

elementy listy podajemy w nawiasach kwadratowych

```
>>> lista = ["komputer", "liczba", 3.14, 26, False]
>>> lista[0]
'komputer'
>>> lista[3]
26
>>> type(lista)
<class 'list'>
```

dostęp do poszczególnych elementów listy uzyskujemy dzięki indeksowi (kolejnym numerze elementu z listy, zaczynając od zera)

https://www.w3schools.com/python/python_lists.asp

List (lista) - właściwości

```
>>> lista
['komputer', 'liczba', 3.14, 26, False, 19]
```

nasza lista

```
>>> lista.append(19)
>>> lista
['komputer', 'liczba', 3.14, 26, False, 19]
```

dołożenie elementu na koniec listy

```
>>> lista[-1]
19
>>> lista[-2]
False
>>> lista[-6]
'komputer'
```

indeksy ujemne zwracają elementy od końca listy

```
>>> lista[0:5]
['komputer', 'liczba', 3.14, 26, False]
>>> lista[4:5]
[False]
>>> lista[1:3]
['liczba', 3.14]
```

zakres zwraca elementy w zakresie listy

```
>>> lista.index(3.14)
2
>>> lista.index(19)
5
>>> lista.index("komputer")
0
```

lista zwraca indeks danego elementu

```
>>> 20 in lista
False
>>> 'komputer' in lista
True
```

sprawdzenie, czy dany element jest na liście

List (lista) – uwaga na kopiowanie

```
>>> a = [1,2,3,4]
>>> b = a
>>> b
[1, 2, 3, 4]
>>> a[0] = "jeden"
>>> a
['jeden', 2, 3, 4]
>>> b
['jeden', 2, 3, 4]
```

nasza lista o nazwie a

przypisujemy naszą listę do listy b
(to nie jest kopiowanie - **lista a oraz b wskazują ten sam fragment pamięci**)

zmieniamy element na naszej liście

zmiana elementu na naszej liście
pociąga za sobą taką samą zmianę w liście b

List (lista) – uwaga na kopiowanie

```
>>> a
['jeden', 2, 3, 4]
>>> b = a[:]
>>> b
['jeden', 2, 3, 4]
>>> a[1] = "dwa"
>>> a
['jeden', 'dwa', 3, 4]
>>> b
['jeden', 2, 3, 4]
```

nasza lista o nazwie a

kopiujemy cały zakres listy a do listy b

zmieniamy drugi element w naszej liście

drugi element w kopii listy został niezmieniony

shellow copy (kpiuje referencje obiektów, a nie same obiekty, w przeciwieństwie do *deep copy*)

Tuple (krotka)

uporządkowana, **niezmienna** kolekcja danych różnych typów

elementy krotki podajemy w nawiasach okrągłych

dostęp do elementów krotki uzyskujemy dzięki indeksowi

```
>>> mojaTupla = (1, "dwa", True)
>>> mojaTupla[1]
'dwa'
>>> mojaTupla[0] = "jeden"
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
```

nie można zmienić elementu krotki

https://www.w3schools.com/python/python_tuples.asp

Set

nieuporządkowana, niezmienna, nieindeksowana kolekcja danych różnych typów
elementy nie mogą się duplikować

elementy set podajemy w nawiasach klamrowych

```
>>> mojSet = { 1, 2, "trzy"}
>>> mojSet
{1, 2, 'trzy'}
>>> mojSet[0]
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: 'set' object is not subscriptable
```

w set nie można zmienić elementu, można go usunąć i dodać inny

w set nie ma indeksu

https://www.w3schools.com/python/python_sets.asp

od wersji Pythona 3.7 w górę

Dictionary (słownik)

uporządkowana, zmienna, nieindeksowana kolekcja par danych klucz:wartość
elementy nie mogą się duplikować (nie mogą być 2 elementy o tym samym kluczu)

```
>>> myDictionary = {
... "kot" : "cat",
... "dom" : "home",
... "rok" : 2023
... }
```

słownik to kolekcja par key:value

```
>>> myDictionary
{'kot': 'cat', 'dom': 'home', 'rok': 2023}
```

```
>>> myDictionary["kot"]
'cat'
```

podając klucz możemy dostać jego wartość

Operatory arytmetyczne dwuargumentowe

wykonaj w konsoli

wynik

+	dodawanie	*print(4+5)*	*9*
-	odejmowanie	*print(4-8)*	*-4*
*	mnożenie	*print(3*2)*	*6*
**	potęgowanie	*print(3**2)*	*9*
/	dzielenie	*print(3/2)*	*1.5*
//	dzielenie całkowite np.: 10/4=2 (dzielimy 2 liczby – iloraz jest zaokrąglany do liczby całkowitej)	*print(3//2)*	*1*
%	modulo – reszta z dzielenia liczb całkowitych	*print(9%4)*	*1*

to samo co 3*3

9 / 4 = 2 reszty 1

https://www.w3schools.com/python/python_operators.asp

Operatory relacji (porównania)

wykonaj w konsoli

==	równy	`>>> 3==4` `False`	`>>> 10==10` `True`
!=	różny	`>>> 3!=4` `True`	`>>> 13!=13` `False`
>	większy	`>>> 12>3` `True`	`>>> 2>30` `False`
<	mniejszy	`>>> 2<4` `True`	`>>> 12<3` `False`
>=	większy lub równy	`>>> 12>=12` `True`	`>>> 1>=12` `False`
<=	mniejszy lub równy	`>>> 30<=30` `True`	`>>> 30<=3` `False`

https://www.w3schools.com/python/python_operators.asp

Instrukcja warunkowa

```
if warunek:
    instrukcja
```

jeśli spełniony jest *warunek* (ma wartość True)
wykonywana jest *instrukcja*

```
if warunek:
    instrukcja
else:
    instrukcja2
```

jeśli spełniony jest *warunek* (ma wartość True)
wykonywana jest *instrukcja*
w innym przypadku wykonywana jest *instrukcja2*

```
if warunek:
    instrukcja
elif warunek2:
    instrukcja2
else:
    instrukcja3
```

jeśli spełniony jest *warunek* (ma wartość True)
wykonywana jest *instrukcja*
w innym przypadku jeśli spełniony jest *warunek2*
(ma wartość True) wykonywana jest *instrukcja2*
w innym przypadku wykonywana jest *instrukcja3*

Instrukcja warunkowa

program *instrukcjaWarunkowa.py*

```
a = 100
b = 10
if a > b:
    print("liczba " + str(a) + " jest większa od liczby " + str(b))
elif a==b:
    print("liczby są równe")
else:
    print("liczba " + str(b) + " jest większa od liczby " + str(a))
```



```
liczba 100 jest większa od liczby 10
```

Instrukcja warunkowa

program *instrukcjaWarunkowa1.py*

```
# input zwraca string
a = input("podaj pierwszą liczbę całkowitą: ")
b = input("podaj drugą liczbę całkowitą: ")
a = int(a)  # zmiana ciągu tekstowego na liczbę całkowitą
b = int(b)  # zmiana ciągu tekstowego na liczbę całkowitą
if a > b:
    print("liczba " + str(a) + " jest wieksza od liczby " + str(b))
elif b > a:
    print("liczba " + str(b) + " jest wieksza od liczby " + str(a))
else:
    print("liczby są równe")
```

konkatenacja (sklejanie) stringów

zamiana liczby na string

Pętla for

kolejne słowo zaczyna się od nowej linii

```
for.py > ...
1   for i in range(5):
2       print("pionowo")
3
4   for i in range(5):
5       print("poziomo ", end="")
6
7   print("") # enter
8
9   for i in range(3,6):
10      print(i)
11
12  for i in range(2,10,2):
13      print(str(i) + " ", end="")
```

zakres

printowanie w poziomie

od do

od do krok

```
pionowo
pionowo
pionowo
pionowo
pionowo
poziomo poziomo poziomo poziomo poziomo
3
4
5
2 4 6 8
```

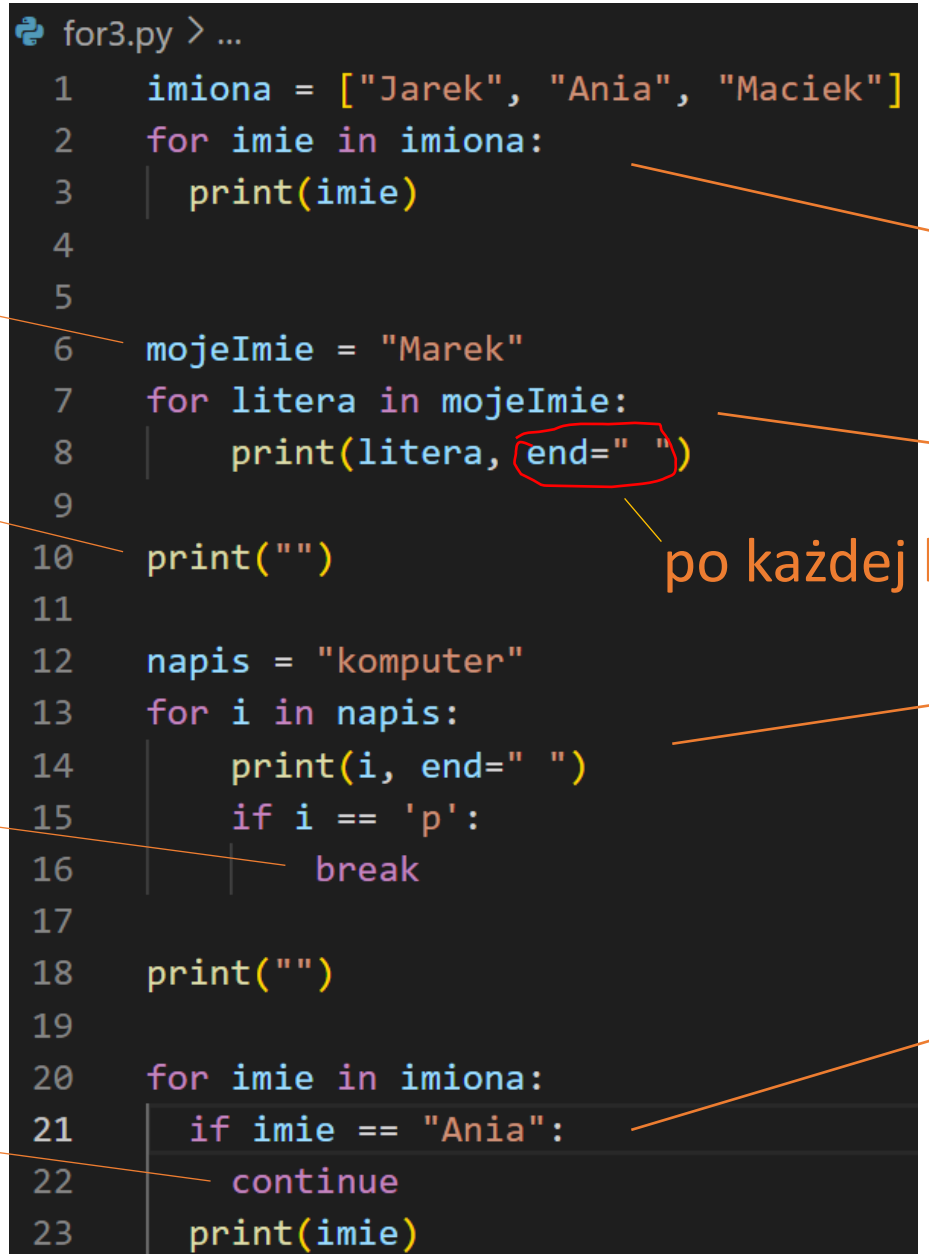
kolejne słowa po kolei

liczby parzyste

spacja pomiędzy liczbami

https://www.w3schools.com/python/python_for_loops.asp

Pętla for



tablica imion

ciąg tekstowy
(w cudzysłowach)
jest tablicą

znak nowego wiersza

po każdej literze spacja

break przerywa pętlę

continue wraca
do początku pętli

```
Jarek
Ania
Maciek
```

```
M a r e k
```

```
k o m p
```

```
Jarek
Maciek
```

Pętla for

for2.py > ...

```
1   for x in range(5):
2       print(x)
3   else:
4       print("Skończone!")
5
6
7
8   print("owocowy wierszyk :)")
9
10  opisy = ["rosną ", "coraz większe ", "smacznie zajadam "]
11  owoce = ["gruszki", "jabłka", "śliwki"]
12
13  for opis in opisy:
14      for owoc in owoce:
15          print(opis, owoc)
16
17
18  klasa = [1,2,3]
19  for x in klasa:
20      pass
```

```
0
1
2
3
4
Skończone!
```

po skończonej pętli wypisuje się napis ”Skończone!”

```
owocowy wierszyk :)
rosną  gruszki
rosną  jabłka
rosną  śliwki
coraz większe  gruszki
coraz większe  jabłka
coraz większe  śliwki
smacznie zajadam  gruszki
smacznie zajadam  jabłka
smacznie zajadam  śliwki
```

pętle zagnieżdżone

pętla nie może być pusta, dzięki pass (w pustej pętli) nie drukują się błędy

Pętla while

```
while.py > ...
1   i = 0
2   while i < 10:
3       print(i)
4       i += 1
5
6   k = 0
7   while k < 2000000:
8       print(k, end=" ")
9       k += 2
10      if k == 10:
11          break
12
13
14  print("")
15
16  m = 0
17  while m < 20:
18      m += 1
19      if m % 2 == 0:
20          continue
21      print(m, end=" ")
```

```
0
1
2
3
4
5
6
7
8
9
```

licznik pętli

break przerywa pętlę

```
0 2 4 6 8
```

co druga liczba

liczby nieparzyste

continue wraca
do początku pętli

```
1 3 5 7 9 11 13 15 17 19
```

https://www.w3schools.com/python/python_while_loops.asp

funkcja

definicja funkcji o nazwie poleKwadratu

nazwa funkcji

parametr funkcji *(w nawiasach)*

słowo kluczowe def
(z angielskiego definition)

```
def poleKwadratu(a):
    return a**2


pole = poleKwadratu(2)
print(pole)  # 4
```

wartość zwracana przez funkcję
(zmienna a do potęgi drugiej)

zmienna lokalna pole

komentarz (program tego nie widzi)

wartość zwracana przez funkcję zostaje przypisana do zmiennej pole

funkcja

2. funkcja liczy potęgę zmiennej a (dwa do potęgi drugiej wynosi 4) i zwraca wynik

2

```
def poleKwadratu(a):
    return a**2

3                   1
pole = poleKwadratu(2)
print(pole)  # 4
```

4

1 2 3 4

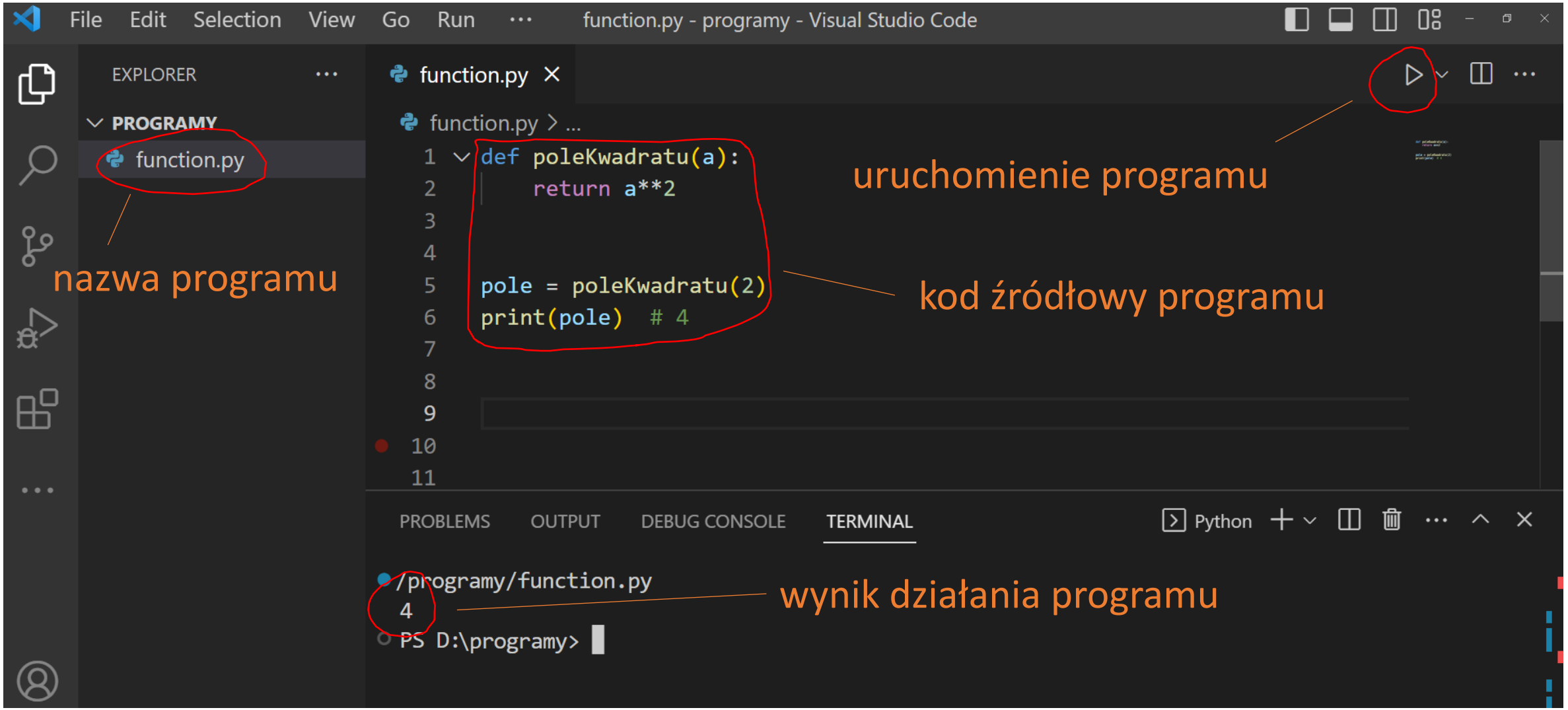
kolejność działania programu

3. wartość zwrócona przez funkcję (liczba 4) jest przypisana do zmiennej pole

1. wywołanie funkcji z argumentem 2, który zostaje skopiowany do zmiennej a (parametru funkcji)

4. zawartość zmiennej pole jest wypisana w konsoli

funkcja


File Edit Selection View Go Run ···
function.py - programy - Visual Studio Code
EXPLORER
PROGRAMY
function.py
function.py
function.py > ...
1 def poleKwadratu(a):
2 return a**2
3
4
5 pole = poleKwadratu(2)
6 print(pole) # 4
7
8
9
10
11
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
Python
/programy/function.py
4
PS D:\programy>
nazwa programu
uruchomienie programu
kod źródłowy programu
wynik działania programu

funkcja

zdefiniować następujące funkcje i je wywołać z argumentami:

a = 3

b = 4

h = 5

```
poleProstakata(a,b)
poleTrojkata(a,h)
```

szyfr Cezara

cezar.py > ...

```python
# szyfrowanie, bez polskich znaków

klucz = 1

def zaszyfruj(slowo, klucz):
    for znak in slowo:
        print(znak + ": " + str(ord(znak))) # kod ASCII
    print("kod ASCII 98: " + chr(98)) # znak podanego kodu ASCII
    zaszyfrowane = ""
    for znak in slowo:
       zaszyfrowane += chr((ord(znak) - 97 + klucz) % 26 + 97)
    return zaszyfrowane

def odszyfruj(zaszyfrowane,klucz):
    slowo = zaszyfruj(zaszyfrowane, 26 - klucz) # odszyfrowuje sie
                                                # szyfrując kluczem
                                                # dopełniajacym do 26
    return slowo

slowo = input("podaj słowo do zaszyfrowania: ")

zaszyfrowane = zaszyfruj(slowo, klucz)
odszyfrowane = odszyfruj(zaszyfrowane, klucz)

print("zaszyfrowane: " + zaszyfrowane)
print("odszyfrowane: " + odszyfrowane)
```