Funkcje w Pythonie

Python 4 Beginners

Zadanie 1

Napisz program, który narysuje w konsoli choinkę z gwiazdek przedstawioną poniżej:

daftcode

Zadanie 2

Napisz program, który narysuje trzy choinki z gwiazdek, jedna pod drugą

```
  *
 ***
*****
  *
 ***
*****
  *
 ***
*****
```

Funkcje

```
def hello():
    print('Hello World!')
```

```
>>> hello()
Hello World!
```

daftcode

Zagadka

```
n = 100
def change_n():
    n = 200

change_n()
print(n)
```

Zasięg widoczności

```
n = 100  # tworzę globalną dla całego pliku zmienną o nazwie n
def change_n():
    n = 200  # tworzę lokalną dla funkcji zmienną o nazwie n

change_n()
print(n)  # wypisuję globalne n=100
```

global

```
n = 100
def change_n():
    global n  # deklaruję, że planuję zmieniać globalne n
    n = 200  # wpisuję wartość do globalnego n

change_n()
print(n)  # wypisuję globalne n=200
```

Ostrożnie z nazwami!

```
print = 5
print('Hello World!')
```

```
TypeError: 'int' object is not callable
```

Zadanie 3

Dodaj do programu możliwość rysowania choinek różnej wysokości.

Argumenty

```
def is_bigger(a, b):
    if a > b:
        print('Bigger')
    else:
        print('Not bigger')
```

```
>>> is_bigger(3, 1)
Bigger
>>> is_bigger(3, 1)
Bigger
```

Zadanie 4

Dodaj do programu możliwość zmiany rysowanego symbolu.

Nazwane argumenty

```
def is_bigger(a, b):
    if a > b:
        print('Bigger')
    else:
        print('Not bigger')
```

```
>>> is_bigger(3, 1)
Bigger
>>> is_bigger(a=3, b=1)
Bigger
>>> is_bigger(b=3, a=1)
Not bigger
>>> is_bigger(1, b=3)
Not bigger
```

Zadanie 5

Zmodyfikuj dotychczasowy program w taki sposób, by domyślnym rysowanym symbolem było *.

Domyślne argumenty

```
def is_bigger(a, b=5):
    if a > b:
        print('Bigger')
    else:
        print('Not bigger')
```

```
>>> is_bigger(3)
Not bigger
>>> is_bigger(a=3)
Not bigger
>>> is_bigger(3, 1)
Bigger
>>> is_bigger(b=1, a=3)
Bigger
```

Domyślne argumenty typów zmiennych

ŹLE:

```
def print_words(words=[]):
    words.append('always_print')
    for word in words:
        print(word)
```

DOBRZE:

```
def print_words(words=None):
    if words is None:
        words = []
    words.append('always_print')
    for word in words:
        print(word)
```

Zadanie 6

Dodaj do programu zliczanie liczby narysowanych symboli.

Zwracanie wartości

```
def sum_numbers(a, b):
    return a + b
```

```
>>> x = sum_numbers(3, 5)
>>> x
8
```

daftcode

Zwracanie wartości

```
def get_even_or_odd(number):
    if number % 2 == 0:
        return 'even'
    return 'odd'
```

```
>>> x = get_even_or_odd(3)
>>> x
odd
>>> y = get_even_or_odd(4)
>>> y
even
```

daftcode

Zwracanie wartości

```
def hello():
    print('Hello World!')
```

```
>>> x = hello()
Hello World!
>>> print(x)
None
```

Zwracanie wielu wartości

```
def get_sum_and_product(a, b):
    sum = a + b
    product = a * b
    return sum, product
```

```
>>> x, y = get_sum_and_product(3, 5)
>>> print(x)
8
>>> print(y)
15
```

daftcode

Zadanie 7

Zmień program w taki sposób, by jedna funkcja umożliwiała narysowanie kilku choinek.

Przyjmowanie dowolnej liczby argumentów

```
def sum_numbers(*numbers):
    total = 0
    for n in numbers:
        total += n
    return total
```

```
>>> x = sum_numbers(3, 5, 10, 12)
>>> x
30
```

daftcode

Zadanie 8

Zmień program w taki sposób, by każda choinka mogła mieć swoją nazwę, której odpowiada wysokość. W wyniku przed choinką wypisz jej nazwę.

Przyjmowanie dowolnej liczby argumentów

```
def print_kwargs(**kwargs):
    for k, v in kwargs.items():
        print(k)
        print(v)
```

```
>>> print_kwargs(a=3, b=5)
a
3
b
5
```

daftcode

Kolejność argumentów

```
def function(a, b=1, *args, **kwargs):
    print(a)
    print(b)
    print(args)
    print(kwargs)
```

Funkcja to obiekt

```
def my_func(a):
    return a

other_func = my_func
other_func(12)

my_func.some_attr = 5

print(my_func.__name__)
print(other_func.__name__)

my_func = 'nope'
```

Docstring

```
def my_func(a):
    """
    Function that returns its argument
    """
    return a

print(my_func.__doc__)
help(my_func)
```

Funkcje rekurencyjne

```
def fibo(n):
    if n == 0:
        return 0
    elif n == 1:
        return 1
    else:
        return fibo(n - 1) + fibo(n - 2)
```

```
>>> x = fibo(6)
8
```

daftcode

Importy

Import modułu

```
import other_module

x = other_module.some_global_variable
other_module.some_function()
```

Import wybranych atrybutów

```
from other_module import some_function, some_global_variable

x = some_global_variable
some_function()
```