Pystart.pl

Argumenty naszych funkcji

lekcja dwudziesta

PyStart #20 Argumenty naszych funkcji

Omówienie

```python
def calculate_circle_area(r):
    pass


def calculate_triangle_area(a, h):
    pass

```

PyStart #20 Argumenty naszych funkcji

Argumenty domyślne

```
1   def calculate_rectangle_area(a=10, b=20):
2       pass
3
```

```
5   calculate_rectangle_area(20)
6
7   calculate_rectangle_area(20, 100)
8
9   calculate_rectangle_area()
10
```

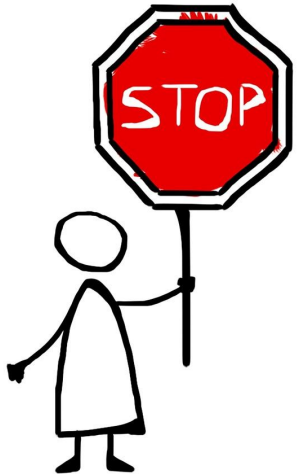
PyStart #20 Wchodzimy w funkcje

Zadania dla nabrania wprawy

20.1

1. Przygotuj funkcję, która będzie szukała wspólnych dzielników dla dwóch liczb. Funkcja posiada trzeci argument domyślny określający wartość początkową, gdzie największy wspólne dzielniki muszą być zawsze większe niż ta wartość.

```
5   calculate_common_divisor(3, 6)
6   # zwróci [3]
7   calculate_common_divisor(3, 6, 4)
8   # zwróci None
9   calculate_common_divisor(16, 8)
10  # zwróci [2, 4, 8]
```



PyStart #20 Argumenty naszych funkcji

Argumenty nazwane

```
1  def calculate_brutto(netto, vat):
2      return netto * (1 + vat)
3
4  calculate_brutto(netto=150, vat=0.23)
```

- ➔ Każdy argument może być podany wraz z nazwą
- ➔ Nie ma znaczenia kolejność argumentów nazwanych
- ➔ W ten sposób łatwiej jest rozkminić czego dotyczy przekazywany argument

PyStart #20 Argumenty naszych funkcji

Argumenty nienazwane i nazwane

```python
def calculate_brutto(quantity, netto, vat=0.23):
    return quantity * netto * (1 + vat)

calculate_brutto(10, netto=150, vat=0.23)
calculate_brutto(10, vat=150, netto=0.23)
```

- ➔ Argument nienazwany idzie na pierwszy ogień
- ➔ Kolejność argumentów nienazwanych musi być zachowana
- ➔ Argumenty nazwane mogą być pomijane jeśli mają wartości domyślne

PyStart #20 Argumenty naszych funkcji

Podsumowanie

- Korzystajmy z argumentów nazwanych by kod był bardziej czytelny lub by pomijać niektóre z argumentów.
- Argumenty z wartością domyślną ułatwiają nam korzystanie z funkcji w przyszłości.
- W przypadku nazywania argumentów kolejność ich przekazywania nie ma żadnego znaczenia. Niemniej warto jest ją zachować dla spójności.

PyStart #20 Argumenty naszych funkcji

Typy argumentów

```python
def add_numbers(a: int, b: int) -> int:
    return a + b


def calculate_length(a: str) -> int:
    return len(a)


def is_adult(age: int) -> bool:
    return age >= 18

```

PyStart #20 Argumenty naszych funkcji

Dwie ciekawostki

```python
from typing import Union

def print_data(something: Union[int, str]):
    print('Czas na wyświetlenie!')
    print(something)

print_data('Kacper')
print_data(123)
print_data(True)
```

PyStart #20 Argumenty naszych funkcji

A co jeśli chcemy zwrócić więcej danych?

```
2   def get_car_details():
3       return {
4           'brand': 'Fiat',
5           'model': '126p',
6           'nickname': 'Maluch'
7       }
8
9   def get_car_size():
10      width = 1377
11      height = 1335
12      length = 3054
13
14      return width, height, length
15
16  car_width, car_height, car_length = get_car_size()
17
```

PyStart #20 Argumenty naszych funkcji

Zadania dla nabrania wprawy

20.2

1. Przygotuj funkcję, która na podstawie czasu pracy, ilości zużywanych kilowatogodzin odpowie ile zapłacę za prąd, który zużywa urządzenie. Domyślny koszt 1 kwh to 0,617.

2. Przygotuj funkcję, której deklaracja będzie wyglądała następująco:

   ```
   def count_numbers(numbers: list, count_odd:bool = True, count_even:bool = True):
   ```

   Zadaniem funkcji będzie odpowiedź na pytanie ile jest liczb spełniających podane w argumentach wymagania w liście przekazanej w pierwszym przedziale.