



# Get/Set metodai, debug

Donatas Noreika

## GET/SET metodai

```
class Darbuotojas:
    def __init__(self, vardas, pavarde,
atlyginimas):
        self.vardas = vardas
        self.pavarde = pavarde
        self.atlyginimas = atlyginimas

domas = Darbuotojas("Domas", "Rutkauskas", 1200)
domas.atlyginimas = 1500
print(domas.atlyginimas)
1500
```

Problema:

```
domas = Darbuotojas("Domas", "Rutkauskas", 1200)
domas.atlyginimas = -150
print(domas.atlyginimas)
-120
```

```
class Darbuotojas:
    def __init__(self, vardas, pavarde,
atlyginimas):
        self.vardas = vardas
        self.pavarde = pavarde
        self.__atlyginimas = atlyginimas

def set_atlyginimas(self, naujas):
    if naujas < 0:
        print("Atlyginimas negali būti
neigiamas")
    else:
        self.__atlyginimas = naujas
```

```
domas = Darbuotojas("Domas", "Rutkauskas", 1200)
domas.set_atlyginimas(-1200)
```

**Atlyginimas negali būti neigiamas**

```
print(domas.atlyginimas)
```

```
AttributeError: 'Darbuotojas' object has no
attribute 'atlyginimas'
```

```
def get_atlyginimas(self):  
    return self.__atlyginimas
```

```
domas = Darbuotojas("Domas", "Rutkauskas", 1200)  
print(domas.get_atlyginimas())  
1200
```

```
domas.atlyginimas = 500  
print(domas.get_atlyginimas())  
1200
```

## Dekinatorius @Property:

```
class Darbuotojas:
    def __init__(self, vardas, pavarde,
atlyginimas):
        self.vardas = vardas
        self.pavarde = pavarde
        self.__atlyginimas = atlyginimas

    @property
    def atlyginimas(self):
        return self.__atlyginimas
```

```
@atlyginimas.setter
def atlyginimas(self, naujas):
    if naujas < 0:
        print("Atlyginimas negali būti
              neigiamas")
    else:
        self.__atlyginimas = naujas
```



```
domas = Darbuotojas("Domas", "Rutkauskas", 1200)
print(domas.atlyginimas)
1200
```

```
domas.atlyginimas = 1500
print(domas.atlyginimas)
1500
```

```
domas.atlyginimas = -150
Atlyginimas negali būti neigiamas
```

```
print(domas.atlyginimas)
1500
```

Kaip ištrinti savybę:

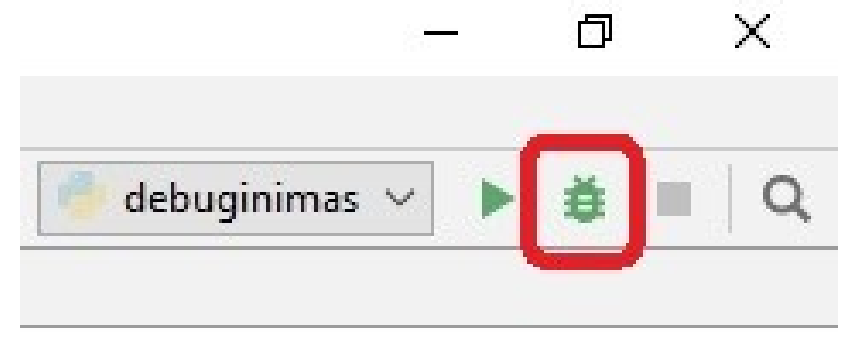
```
@atlyginimas.deleter  
def atlyginimas(self):  
    self.__atlyginimas = 0
```

```
domas = Darbuotojas("Domas", "Rutkauskas", 1200)  
del domas.atlyginimas  
print(domas.atlyginimas)  
0
```

## Klaidų taisymas (Debug)

```
a = 5  
b = 6  
c = a + b  
  
print(c)
```

1	a = 5
2	b = 6
3	c = a + b
4	




```
1 a = 5    a: 5  
2 b = 6    b: 6  
3 c = a + b  c: 11  
4
```

```
5  pass
```

## Variables

### Special Variables

 **a** = {int} 5

 **b** = {int} 6

 **c** = {int} 11

```
sarasas = [3, 15, 454, 78, 92, 16]
naujas_sarasas = []
```

```
for skaicius in sarasas:
    naujas_sarasas.append(skaicius + 2)
```

```
1 sarasas = [3, 15, 454, 78, 92, 16] sarasas: <class 'list'>: [3, 15, 454,
2 naujas_sarasas = [] naujas_sarasas: <class 'list'>: [5, 17, 456, 80, 94]
3
4 for skaicius in sarasas: skaicius: 16
5 naujas_sarasas.append(skaicius + 2)
```

Debug: debuginimas x debuginimas x

Debugger Console

Frames

MainThread

<module>, debuginimas.py:5

execfile, \_pydev\_execfile.py:18

run, pydevd.py:1068

main, pydevd.py:1658

<module>, pydevd.py:1664

Variables

Special Variables

naujas\_sarasas = {list} <class 'list'>: [5, 17, 456, 80, 94]

sarasas = {list} <class 'list'>: [3, 15, 454, 78, 92, 16]

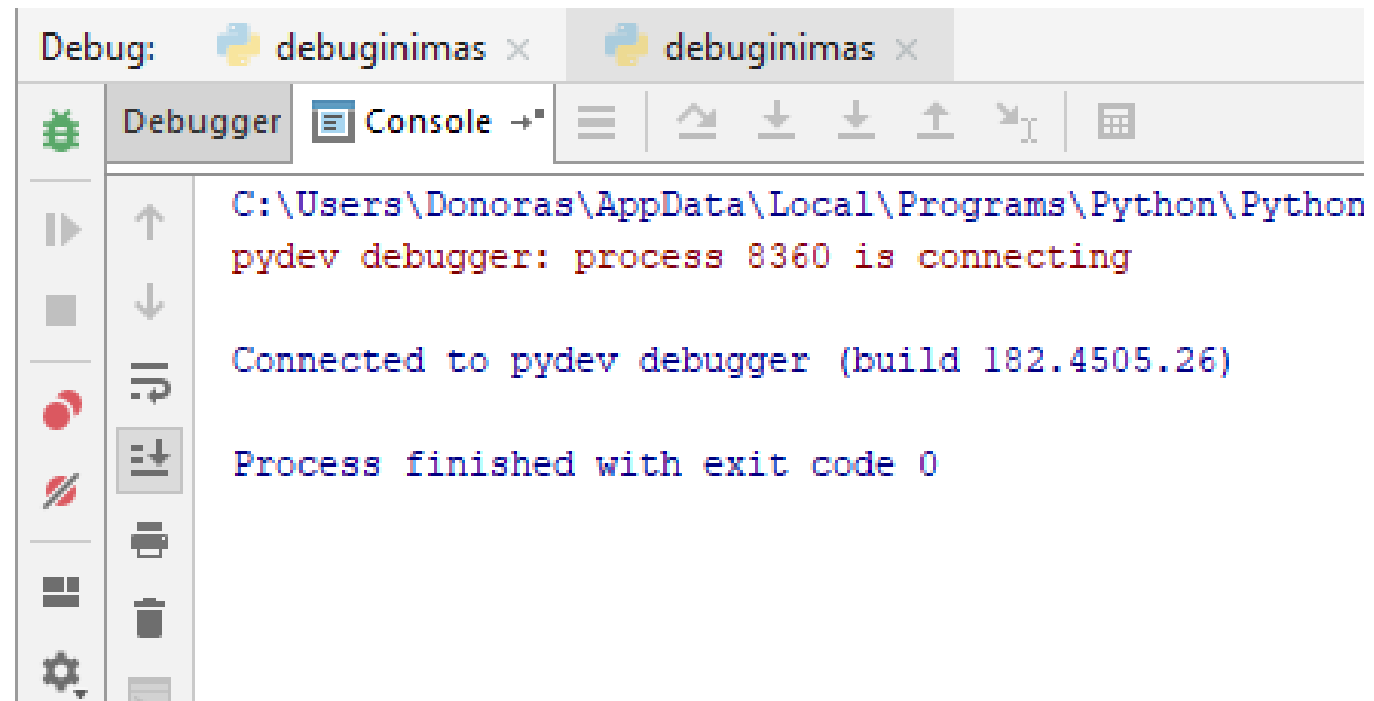
skaicius = {int} 16



```
def patikrinti_skaiciu(skaicius):  
    if skaicius < 0:  
        print("Skaičius teigiamas")  
    if skaicius == 0:  
        print("Skaičius lygus 0")  
    if skaicius < 0:  
        print("Skaičius neigiamas")
```

```
patikrinti_skaiciu(20)
```

```
1  def patikrinti_skaiciu(skaicius):  
2      if skaicius < 0:  
3          print("Skaičius teigiamas")  
4      if skaicius == 0:  
5          print("Skaičius lygus 0")  
6      if skaicius < 0:  
7          print("Skaičius neigiamas")  
8  
9  patikrinti_skaiciu(20)
```




```
def patikrinti_skaiciu(skaicius):  
    if skaicius > 0:  
        print("Skaičius teigiamas")  
    if skaicius == 0:  
        print("Skaičius lygus 0")  
    if skaicius < 0:  
        print("Skaičius neigiamas")
```


```
patikrinti_skaiciu(20)
```

**Skaičius teigiamas**

```
1 class Automobilis():
2     def __init__(self, modelis, metai):
3         self.__modelis = modelis
4         self.__metai = metai
5
6 toyota = Automobilis("Toyota Auris", 2015)  toyota: <
7 tesla = Automobilis("Tesla Y", 2020)
```

## Variables

+ >  Special Variables

▼  `toyota` = {Automobilis} <\_\_main\_\_.Automobilis object at 0x0000023841A36780>

189 `_Automobilis__metai` = {int} 2015

189 `_Automobilis__modelis` = {str} 'Toyota Auris'

## **Užduotis 1 (pagal 6 paskaitos 1 užduotį)**

### **6 paskaitos 1 užduotis – veiksmai su sakiniu**

Perdaryti 6-1 užduotį taip, kad:

- Leistų vartotojui įvesti norimą sakinį
- Sakinio objektui priskyrus įvestą sakinį iš vieno žodžio, parodytų pranešimą, kad sakinytis turi būti bent iš dviejų žodžių
- Paleistų visus objekto metodus

## **Užduotis 2 (pagal 6 paskaitos 2 užduotį)**

**6 paskaitos 2 užduotis – programa, skaičiuojanti, kiek metų, mėnesių, savaičių, dienų, valandų, minučių, sekundžių praėjo nuo įvestos datos**

Perdaryti 6-2 užduotį taip, kad:

- Leistų vartotojui įvesti norimą datą
- Parodytų pranešimą apie neteisingai įvestus metus, jei jie mažesni už 0 arba didesni už 3000
- Parodytų pranešimą apie neteisingai įvestą mėnesį, jei jis mažesnis už 1 arba didesnis už 12



- Parodytų pranešimą apie neteisingai įvestą dieną, jei ji mažesnė už 1 arba didesnė už 31
- Parodytų pranešimą apie neteisingai įvestą valandą, jei ji mažesnė už 0 arba didesnė už 24
- Parodytų pranešimą apie neteisingai įvestas minutes, jei jos mažesnė už 1 arba didesnės už 60

## **Užduotis 3 (pagal 6 paskaitos 3 užduotį)**

### **6 paskaitos 3 užduotis – sukurti automobilio objektą**

Perdaryti 6-3 užduotį taip, kad:

- Leistų vartotojui įvesti automobilio modelį, metus ir kuro tipą
- Parodytų pranešimą apie neteisingai įvestus metus, jei jie mažesni už 1800 arba didesni už 2200
- Atspausdintų informaciją apie automobilį: įvestus metus, modelį bei kuro tipą

## **Užduotis 4 (pagal 6 paskaitos 4 užduotį)**

### **6 paskaitos 4 užduotis – sukurti darbuotojo objektą**

Perdaryti 6-4 užduotį taip, kad:

- Leistų vartotojui įvesti darbuotojo vardą, valandos įkainį bei datą, nuo kurios jis dirba
- Parodytų pranešimą apie neteisingai įvestus metus, jei jie mažesni už 1900
- Parodytų pranešimą apie neteisingai įvestą mėnesį, jei jis mažesnis už 1 arba didesnis už 12
- Parodytų pranešimą apie neteisingai įvestą dieną, jei ji mažesnė už 1 arba didesnė už 31

- Parodytų pranešimą apie neteisingai įvestą datą, jei ji didesnė už šiandienos datą
- Atspausdintų per nurodytą laikotarpį darbuotojo uždirbtą sumą

## **Užduotis 5 (pagal 5 paskaitos 3 užduotį)**

**5 paskaitos 3 užduotis – sukurti metodą, kuris atspausdintų nurodyto režio keliamuosius metus**

Perdaryti 6-4 užduotį taip, kad:

- Anksčiau sukurta funkcija būtų įdėta į klasę
- Parodytų pranešimą apie neteisingai įvestus metus (nuo arba iki), jei jie mažesni už 0 arba didesni už 10000
- Parodytų pranešimą apie neteisingai įvestus metus, jei data NUO yra didesnė už datą IKI
- Atspausdintų nurodyto režio keliamuosius metus

**Pabaiga**