Anotação de Tipos em Definições de Funções

Prof. Alberto Costa Neto

Programação em Python

Função sem Anotação de Tipos

```
1   def calcular_salario(salario, horas_extras, horas_extras_noturnas):  2 usages
2       carga_horaria_mensal = 220
3       adicional_noturno = 1.2
4       adicional_hora_extra = 1.5
5       hora_base = salario / carga_horaria_mensal
6       return salario + \
7              hora_base * horas_extras * adicional_hora_extra + \
8              hora_base * horas_extras_noturnas * adicional_hora_extra * adicional_noturno
9
10  print(calcular_salario( salario: 1512.0,  horas_extras: 10,  horas_extras_noturnas: 5) )
11  print(calcular_salario( salario: 10,  horas_extras: 5,  horas_extras_noturnas: 1512.0) )
```

- Note que na 2º chamada da função calcular_salario o valor do salário está trocado pelo de horas_extras_noturnas, mas nenhum alerta é dado

Erros de Tipos em Funções

- Inversão de ordem de parâmetros na chamada de uma função
- Esperar que a função retorne um valor de um tipo, mas a função retorna valor outro tipo
- Passar um valor de um tipo diferente do esperado em uma chamada de função

Sintaxe de Anotação de Tipos em Funções

- Na lista de parâmetros, podem ser listados os pares de <identificador>:<tipo>, separados entre si por vírgulas.
- Também pode ser informado o tipo de retorno com a sintaxe -> <tipo>, como no exemplo abaixo:

Lista de Parâmetros com Tipos

Tipo de Retorno

```
def calcular_salario (salario:float, extras:int) -> float:
    ...
```

Anotando Tipos em Função

```
1  def calcular_salario(salario: float, horas_extras:int, horas_extras_noturnas:int) -> float:
2      carga_horaria_mensal:int = 220
3      adicional_noturno: float = 1.2
4      adicional_hora_extra: float = 1.5
5      hora_base: float = salario / carga_horaria_mensal
6      return salario + \
7             hora_base * horas_extras * adicional_hora_extra + \
8             hora_base * horas_extras_noturnas * adicional_hora_extra * adicional_noturno
9
10 print(calcular_salario( salario: 1512.0, horas_extras: 10, horas_extras_noturnas: 5))
11 print(calcular_salario( salario: 10, horas_extras: 5, horas_extras_noturnas: 1512.0))
```

⚠ Expected type 'int', got 'float' instead :11

- Na linha 11 foi trocada a ordem dos parâmetros, mas o tipo diferente do esperado gerou um alerta no PyCharm 2025.2 CE

Mais opções Sintáticas em Definições de Funções

- É possível definir um conjunto de tipos por parâmetro com o operador | entre os nomes dos tipos de dados, <identificador>:<tipo1 | tipo2>

A função soma aceita int ou float como argumentos para x e y e retorna int ou float

```
1  def soma(x: float|int, y: float|int) -> float|int:
2      return x + y
3
4  print(soma( x: 10,  y: 11))
5  print(soma( x: 10.5,  y: 12.5))
6  print(soma( x: 12,  y: 23.5))
```

Retorno pode ser int ou float

```
21
23.0
35.5
```

A importância de Anotação de Tipos

- Documentação dos parâmetros e tipos de retorno
- Disponibilidade de informações dos tipos para IDEs
- Detecção antecipada de erros de tipo
- Prevenção de alguns erros, como uma troca de parâmetros em chamadas de funções