

Functies

In dit hoofdstuk gaan we leren over functies , een essentieel onderdeel van elke programmeertaal. Functies helpen je code georganiseerd, herbruikbaar en overzichtelijk te houden. Je leert hoe je standaardfuncties gebruikt en hoe je je eigen functies kunt schrijven.

- Wat zijn Functies?
- Gebruik van Standaardfuncties
- Eigen Functies maken
- Parameters, Argumenten en Return-waarden
- Afronding Functies

Wat zijn Functies?

Doelstellingen

- Begrijpen wat functies zijn en waarom ze belangrijk zijn.
- Inzicht krijgen in de voordelen van het gebruik van functies.
- Leren over de basisstructuur van een functie in Python.

Inhoud

Wat is een functie?

Een functie is een blok code dat alleen wordt uitgevoerd wanneer het wordt aangeroepen. Functies kunnen parameters (inputs) accepteren en kunnen waarden teruggeven. Ze helpen je om je code op te delen in herbruikbare stukken.

Voordelen van functies

- **Herbruikbaarheid:** Vermijd het schrijven van dubbele code.
- **Overzichtelijkheid:** Maak je code leesbaarder en beter georganiseerd.
- **Onderhoudbaarheid:** Makkelijker om je code te onderhouden en aan te passen.

Basisstructuur van een functie

```
def functienaam():  
    # codeblok
```

Voorbeeld:

```
def begroeting():  
    print("Hallo, welkom bij het programma!")
```

Functie aanroepen:

```
begroeting() # Output: Hallo, welkom bij het programma!
```

Activiteiten

- **Oefening:**

- Schrijf een eenvoudige functie `welkom()` die "Welkom, gebruiker!" print.

- **Discussie:**

- Bespreek met je klasgenoten waarom het handig is om functies te gebruiken in programma's.

Tips

- Gebruik duidelijke en beschrijvende namen voor je functies.
- Let op de inspringing; alle code binnen de functie moet ingesprongen zijn.

Gebruik van Standaardfuncties

Doelstellingen

- Kennismaken met ingebouwde standaardfuncties in Python.
- Leren hoe je deze functies kunt toepassen in je code.
- Begrijpen waar je documentatie over standaardfuncties kunt vinden.

Inhoud

Wat zijn standaardfuncties?

Python biedt een reeks ingebouwde functies die je kunt gebruiken zonder ze zelf te definiëren. Ze zijn direct beschikbaar en vereisen geen extra import.

Veelgebruikte standaardfuncties:

- `print()`: Toont output op het scherm.
- `len()`: Geeft de lengte van een object terug (zoals een string of lijst).
- `type()`: Geeft het datatype van een object.
- `input()`: Vraagt om invoer van de gebruiker.
- `int()`, `float()`, `str()`: Converteert waarden naar respectievelijk een integer, float of string.
- `max()`, `min()`: Geeft het maximum of minimum van een reeks waarden.

Voorbeelden:

```
# Gebruik van len()
woord = "Python"
print(len(woord)) # Output: 6

# Gebruik van input()
naam = input("Wat is je naam? ")
print(f"Hallo, {naam}!")
```

Activiteiten

- **Oefening:**

- Vraag de gebruiker om drie getallen in te voeren en gebruik `max()` en `min()` om het grootste en kleinste getal te vinden.

- **Verkennen:**

- Gebruik `help()` in de Python-interpreter om meer te leren over een standaardfunctie, bijvoorbeeld `help(len)`.

Tips

- Raadpleeg de [officiële documentatie](#) voor een volledige lijst van standaardfuncties.
- Experimenteer met verschillende functies om te ontdekken wat ze doen.

Eigen Functies maken

Doelstellingen

- Leren hoe je je eigen functies kunt definiëren.
- Begrijpen hoe je functies kunt gebruiken om je code te structureren.
- Functies schrijven die specifieke taken uitvoeren.

Inhoud

Een functie definiëren

Gebruik het sleutelwoord `def`, gevolgd door de functienaam en ronde haakjes `()`.

```
def functienaam():  
    # codeblok
```

Voorbeeld zonder parameters:

```
def toon_menu():  
    print("1. Optellen")  
    print("2. Aftrekken")  
    print("3. Vermenigvuldigen")  
    print("4. Delen")
```

Functie aanroepen:

```
toon_menu()
```

Activiteiten

- **Oefening:**
 - Schrijf een functie `lijn()` die een lijn van 30 streepjes `-` print.
- **Praktijk:**
 - Maak een programma dat een begroeting toont, een menu presenteert en de gebruiker om een keuze vraagt, waarbij elke stap in een aparte functie is geplaatst.

Tips

- Definieer je functies bovenaan je script voordat je ze aanroept.
- Houd je functies kort en gericht op één taak voor betere leesbaarheid.

Parameters, Argumenten en Return-waarden

Doelstellingen

- Begrijpen hoe je functies flexibeler maakt met parameters.
- Leren hoe je functies waarden kunt laten teruggeven met `return`.
- Toepassen van deze concepten in je eigen code.

Inhoud

Parameters en Argumenten

- **Parameter:** Een variabele in de functie-definitie die een waarde verwacht.
- **Argument:** De daadwerkelijke waarde die je bij het aanroepen aan de functie doorgeeft.

Voorbeeld met parameters:

```
def begroet(naam):  
    print(f"Hallo, {naam}!")
```

Functie aanroepen met argument:

```
begroet("Sophie") # Output: Hallo, Sophie!
```

Functie met meerdere argumenten:

```
def vermenigvuldig(a, b):  
    resultaat = a * b  
    print(f"Het resultaat is: {resultaat}")
```

```
vermenigvuldig(5, 3) # Output: Het resultaat is: 15
```

Return-waarden

Een functie kan een waarde teruggeven met het `return` sleutelwoord.

```
def oppervlakte_cirkel(r):  
    return 3.1415 * r ** 2  
  
opp = oppervlakte_cirkel(5)  
print(f"De oppervlakte is: {opp}")
```

Activiteiten

- **Oefening:**

- Schrijf een functie `is_even(getal)` die `True` teruggeeft als het getal even is en `False` als het oneven is.

- **Challenge:**

- Maak een functie `bereken_btw(bedrag, percentage)` die de BTW berekent over een gegeven bedrag en percentage.

Tips

- Functies kunnen meerdere parameters hebben; scheid ze met komma's.
- Je kunt meerdere waarden teruggeven met `return`, bijvoorbeeld `return a, b`.

Afronding Functies

In dit hoofdstuk heb je geleerd over:

- **Wat functies zijn en waarom ze belangrijk zijn.**
 - **Het gebruik van standaardfuncties in Python.**
 - **Het definiëren en aanroepen van je eigen functies.**
 - **Het gebruik van parameters en return-waarden om functies flexibeler te maken.**
-

Huiswerk / Opdrachten

- **Oefeningen:**
 - Maak alle opdrachten uit dit hoofdstuk.
- **Programmeeroefeningen:**
 1. Schrijf een functie `omkeren(tekst)` die een gegeven tekst omdraait.
 2. Maak een programma met functies om de omtrek en oppervlakte van geometrische vormen te berekenen (bijv. cirkel, rechthoek, driehoek).

Extra Bronnen

Online Tutorials:

- [Python Functies](#) - W3Schools

Oefensites:

- [HackerRank Python Functies](#)

Tips voor Succes

- **Modulariseer je code:**
 - Door functies te gebruiken, kun je je code beter organiseren en begrijpelijker maken.
- **Gebruik docstrings:**
 - Voeg onder de functie-definitie een korte beschrijving toe tussen drievoudige aanhalingstekens.

```
def groet(naam):  
    """Print een begroeting voor de gegeven naam."""  
    print(f"Hallo, {naam}!")
```

- **Test je functies grondig:**

- Probeer je functies met verschillende inputs om te zien of ze correct werken.

- **Leesbaarheid:**

- Houd je code netjes en goed ingesprongen om fouten te voorkomen en leesbaarheid te bevorderen.