DELEN

We zagen dat je getallen deelt met de operator **/**. Dat geeft als resultaat altijd een float:

```
>>> 7/3
2.3333333333333335
```

Maar je hebt ook de operator **//** waarmee je een gehele deling uitvoert. Dat wil zeggen dat het deel na de komma wordt genegeerd en je als resultaat een int krijgt:

```
>>> 7//3
2
```

Je kunt ook de rest van de deling door een getal opvragen, namelijk met de operator **%**:

```
>>> 7%3
1
```

En dat klopt, want als we de twee vorige berekeningen samennemen, kunnen we eenvoudig narekenen dat 7 gelijk is aan 3*2+1.

Overigens werken de operatoren **//** en **%** ook voor niet-gehele getallen:

```
>>> 7.5/2.1
3.571428571428571
>>> 7.5//2.1
3.0
>>> 7.5%2.1
1.1999999999999997
```

TYPES

Je hebt nu kennisgemaakt met de twee typen getallen: int en float. Het type van een getal of een berekening kun je eenvoudig opvragen:

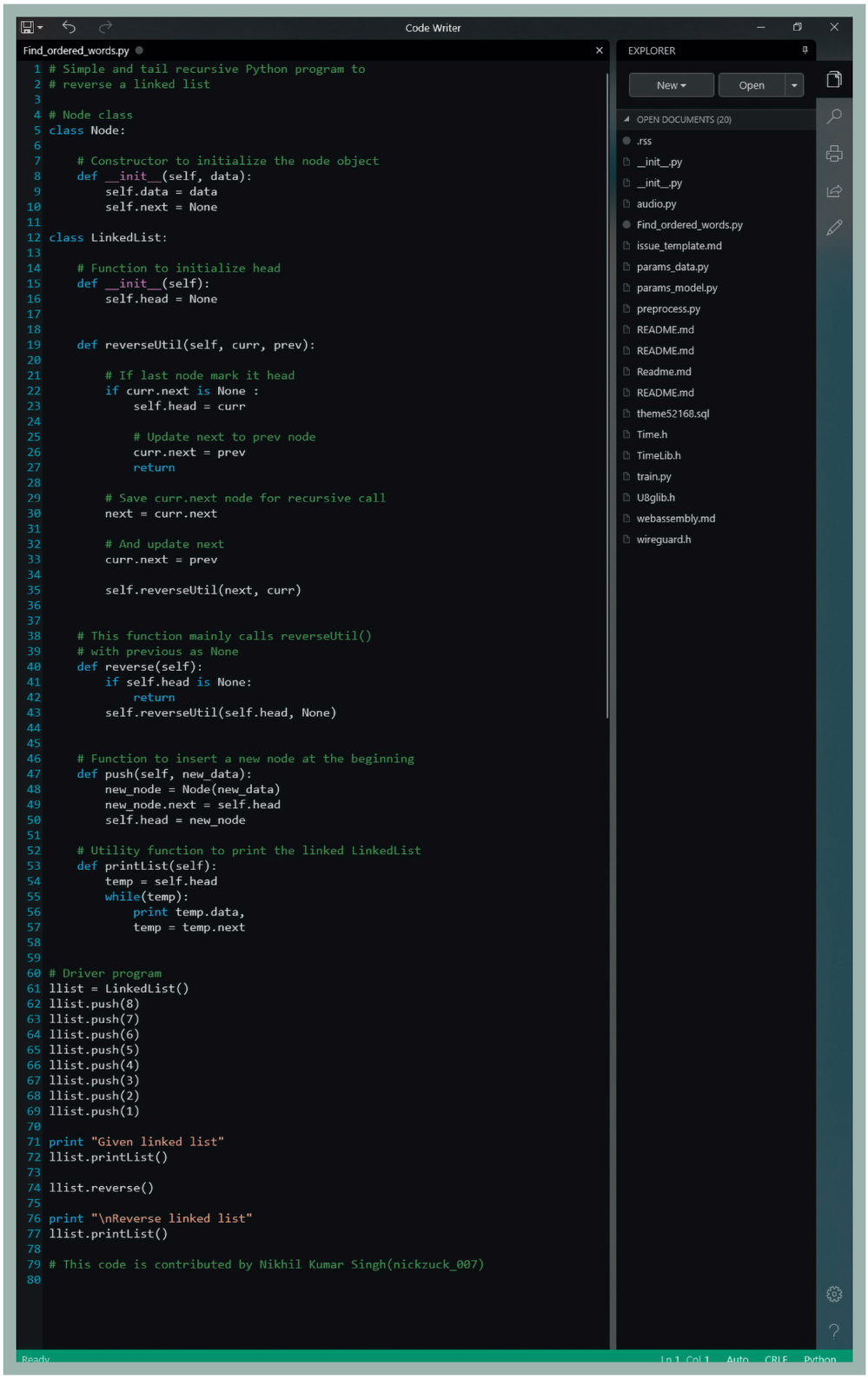
```
>>> type(1+1)
<class 'int'>
>>> type(7.5)
<class 'float'>
>>> type(7.5//2.1)
<class 'float'>
>>> type(7//3)
<class 'int'>
```

Merk op: het resultaat van **7.5//2.1** is **3.0**, wat een float is.

Je kunt getallen ook omzetten van het ene naar het andere type:

```
>>> float(1+1)
2.0
>>> int(2.5)
2
```

Als je een int naar float omzet, dan blijft de waarde van het getal hetzelfde; zet je een float naar int om, dan wordt de waarde na de komma afgebroken.




```
# Simple and tail recursive Python program to
# reverse a linked list

# Node class
class Node:

    # Constructor to initialize the node object
    def __init__(self, data):
        self.data = data
        self.next = None

class LinkedList:

    # Function to initialize head
    def __init__(self):
        self.head = None


    def reverseUtil(self, curr, prev):

        # If last node mark it head
        if curr.next is None :
            self.head = curr

            # Update next to prev node
            curr.next = prev
            return

        # Save curr.next node for recursive call
        next = curr.next

        # And update next
        curr.next = prev

        self.reverseUtil(next, curr)


    # This function mainly calls reverseUtil()
    # with previous as None
    def reverse(self):
        if self.head is None:
            return
        self.reverseUtil(self.head, None)


    # Function to insert a new node at the beginning
    def push(self, new_data):
        new_node = Node(new_data)
        new_node.next = self.head
        self.head = new_node

    # Utility function to print the linked LinkedList
    def printList(self):
        temp = self.head
        while(temp):
            print temp.data,
            temp = temp.next


# Driver program
llist = LinkedList()
llist.push(8)
llist.push(7)
llist.push(6)
llist.push(5)
llist.push(4)
llist.push(3)
llist.push(2)
llist.push(1)

print "Given linked list"
llist.printList()

llist.reverse()

print "\nReverse linked list"
llist.printList()

# This code is contributed by Nikhil Kumar Singh(nickzuck_007)
```

VARIABELEN

Als je de Python-interpreter als rekenmachine gebruikt, wil je misschien de vorige waarde als onderdeel van een volgende berekening gebruiken zonder dat je die helemaal opnieuw hoeft in te typen. Dat kan eenvoudig met de variabele _:

```
>>> 7/3
2.3333333333333335
>>> _*5
11.666666666666668
```

Een variabele is een naam die je aan een waarde geeft. De Python-interpreter kent automatisch de laatste waarde toe aan de variabele _. Maar je kunt ook zelf variabelen aanmaken met een willekeurige