Exercise Book 1

Covering the materials of Chapters 1-4.
Topics: control structures, user input, exception handling, random generation, lists, function definition

Task 1: Armstrong numbers

Produce all *Armstrong numbers* smaller than N. The value of N is given by the user. Validate the user input!

A number is an *Armstrong number* (https://en.wikipedia.org/wiki/Narcissistic_number), if it is the sum of its own digits, each raised to the power of the number of digits. For example 153 is an *Armstrong number*, because $1^3 + 5^3 + 3^3 = 153$. The first few *Armstrong numbers* are: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 153, 370, 371, 407, 1634, etc.

In [1]:

```
valid_input = False
while not valid_input:
    try:
        N = int(input("N := "))
        valid_input = True
    except:
        print("That is not a number.")

print('Armstrong numbers smaller than %d:' % N)
for number in range(0, N):
    orig_number = number
    digits = len(str(number))
    result = 0
    while number > 0:
        last_digit = number % 10
        number = number // 10
        result += last_digit ** digits
    if result == orig_number:
        print(orig_number)
```

```
Armstrong numbers smaller than 10000:
0
1
2
3
4
5
6
7
8
9
153
370
371
407
1634
8208
9474
```

Task 2: Perfect numbers

Produce the first N *Perfect numbers*. The value of N is given by the user. Validate the user input!

In number theory, a *perfect number* (https://en.wikipedia.org/wiki/Perfect_number) is a positive integer that is equal to the sum of its positive divisors, excluding the number itself. For instance, 6 has divisors 1, 2 and 3 (excluding itself), and $1 + 2 + 3 = 6$, so 6 is a *perfect number*. The first few *perfect numbers* are: 6, 28, 496, etc.

In [2]:

```python
valid_input = False
while not valid_input:
    try:
        N = int(input("N := "))
        valid_input = True
    except:
        print("That is not a number.")

print('The first %d perfect numbers:' % N)
found_numbers = 0
number = 1
while found_numbers < N:
    result = 0
    for div in range(1, number):
        if number % div == 0:
            result += div
    if result == number:
        print(number)
        found_numbers += 1
    number += 1
```

```
The first 4 perfect numbers:
6
28
496
8128
```

Task 3: Greatest common divisor

Calculate the *greatest common divisor* of 2 numbers!

Request 2 integer numbers from the user and calculate their greatest common divisor. E.g. for 30 and 105 their greatest common divisor is 15. Do not use the `math.gcd()` built-in function to solve the task.

Hint: use the Euclidean algorithm (https://en.wikipedia.org/wiki/Euclidean_algorithm)

```
In [3]:
```

```
def gcd(a, b):
    while a != b:
        if a > b:
            a -= b
        else:
            b -= a
    return a

num1 = int(input('First number: '))
num2 = int(input('Second number: '))

print('The greatest common divisor of %d and %d is %d' %(num1, num2, gcd(num1, n
um2)))
```

```
The greatest common divisor of 30 and 105 is 15
```

Task 4: Rock–paper–scissors

Implement the popular *rock–paper–scissors* (https://en.wikipedia.org/wiki/Rock%E2%80%93paper%E2%80%93scissors) game, where the user can play against the computer! The human player can type in *rock*, *paper* or *scissors*. Handle incorrect input and request the input again if it does not match one of the three previous options. The computer player randomly chooses one of the options. The game finishes when one of the players won. (It continues with another round upon a draw.)

In [4]:

```
import random
options = ['rock', 'paper', 'scissors']
computer = random.choice(options)

# Read user input with validation
def read_user_input():
    user_input = input("Please type in 'rock', 'paper' or 'scissors': ")
    while not user_input in options:
        print("There must have been a typo. Please try again: ")
        user_input = input("Please type in 'rock', 'paper' or 'scissors': ")
    return user_input

user_input = read_user_input()
while user_input == computer:
    print("There is a tie!")
    computer = random.choice(options)
    user_input = read_user_input()

if user_input == "rock":
    if computer == "paper":
        print ("Paper beats rock. The computer won.")
    else:
        print("Rock beats scissors. You won!")
elif user_input == "paper":
    if computer == "rock":
        print("Paper beats rock. You won!")
    else:
        print("Scissors beat paper. The computer won.")
elif user_input == "scissors":
    if computer == "rock":
        print("Rock beats scissors. The computer won.")
    else:
        print ("Scissors beats paper. You won.")
```

```
Rock beats scissors. You won!
```

Task 5: Guess a number

Write a program which can think a of number between 1 and 100, randomly. The task of the user is to guess that number. In each round the user can make a guess, and the program replies whether the guess is correct or it was too small or too large. The game ends when the user succefully guesses the number.

In [5]:

```python
import random
number = random.randint(1, 100)

guess = int(input("Guess my number between 1 and 100: "))
while guess != number:
    if guess > number:
        print("Your number is too large. Try again. ")
        guess = int(input("Guess a number: "))
    elif guess < number:
        print("Your number is too small. Try again.")
        guess = int(input("Guess a number: "))
else:
    print("This is the correct number!")
```

```
Your number is too large. Try again.

Your number is too small. Try again.

Your number is too small. Try again.

Your number is too large. Try again.

Your number is too small. Try again.

This is the correct number!
```

Question: if the human player is smart, what is the minimum number of guesses, which is always enough?

Task 6: Separation by parity

Given a *list* of numbers, write a program, which separates the odd and even integers in separate lists. E.g.:

```
Input: [45, 83, 90, 11, 24, 98, 87, 39, 9, 6]
Even numbers: [90, 24, 98, 6]
Odd numbers: [45, 83, 11, 87, 39, 9]
```

Here is a list of 20 random numbers between 1 and 100:

In [6]:

```python
import random

numbers = []
for i in range(20):
    numbers.append(random.randint(1, 100))
print(numbers)
```

```
[68, 48, 82, 81, 37, 41, 96, 8, 95, 96, 97, 20, 54, 52, 28, 17, 76,
54, 6, 23]
```

Now write a program which separates them:

In [7]:

```
even_numbers=[]
odd_numbers=[]
for i in numbers:
    if i%2==0:
        even_numbers.append(i)
    else:
        odd_numbers.append(i)
print('The even numbers are: %s' % even_numbers)
print('The odd numbers are: %s' % odd_numbers)
```

```
The even numbers are: [68, 48, 82, 96, 8, 96, 20, 54, 52, 28, 76, 5
4, 6]
The odd numbers are: [81, 37, 41, 95, 97, 17, 23]
```

Task 7: Pyramid

Write the `pyramid(height)` function, which displays a pattern like a pyramid with an asterisk. The height of the pyramid can be defined by the user.

E.g. for $height = 4$, the pyramid would look like:

```
      *
     ***
    *****
   *******
```

In [8]:

```
def pyramid(height):
    for i in range(height):
        print(' '*(height-i-1) + '*'*(2*i+1))

h=int(input('Height of the pyramid: '))
pyramid(h)
```

```
     *
    ***
   *****
  *******
 *********
***********
```

Task 8: Collatz sequence

Write a function which produces the *Collatz sequence* and returns it as a list. The function receives the starting value for the sequence. Request the starting value from the user, call the function and display the generated *Collatz sequence*.

The *Collatz sequence* (https://en.wikipedia.org/wiki/Collatz_conjecture) has a starting value and the next item of the sequence is always calculated from the previous one, defined as follows:

- if the number is even, divide it by two;
- if the number is odd, triple it and add one.

More formally: $f(n) = \begin{cases} n/2 & n \equiv 0 \pmod 2 \\ 3n+1 & n \equiv 1 \pmod 2 \end{cases}$

The sequence stops upon reaching 1. For instance, starting with $n = 12$, one gets the sequence 12, 6, 3, 10, 5, 16, 8, 4, 2, 1.

In [9]:

```
def collatz(number):
    sequence=[number]
    while number>1:
        if number%2==0:
            number=number//2
            sequence.append(number)
        else:
            number=3*number+1
            sequence.append(number)
    return sequence

number=int(input('Starting number of Collatz sequence: '))
print(collatz(number))
```

```
[123, 370, 185, 556, 278, 139, 418, 209, 628, 314, 157, 472, 236, 11
8, 59, 178, 89, 268, 134, 67, 202, 101, 304, 152, 76, 38, 19, 58, 2
9, 88, 44, 22, 11, 34, 17, 52, 26, 13, 40, 20, 10, 5, 16, 8, 4, 2,
1]
```

Task 9: Anagram

An anagram is a word or phrase formed by rearranging the letters of a different word or phrase. For example the words *spear* and *pears* are anagrams. Write a function, which decides whether two words are anagrams or not and returns a boolean value accordingly (*True* or *False*).

Hint: pay attention that a single letter may accour multiple times in a word.

In [11]:

```
def is_anagram(word1, word2):
    # If the size does not match, they cannot be anagrams
    if len(word1) != len(word2):
        return False

    # Check for each character in word1 whether it is also in word2
    for ch in word1:
        if not ch in word2:
            return False
    return True

a=input('First word: ')
b=input('Second word: ')
print('Are they anagrams? %s' % is_anagram(a, b))
```

```
Are they anagrams? True
```

Task 10: Circularly identical lists

A list is called *circular* if we consider the first element as next of the last element. *Circularly identical* lists contain the same elements in the same order, given that two lists that can be obtained from each other if one or more of the elements in one of the lists are rotated/displaced from their original index and placed at the beginning. E.g. the lists `[10, 20, 30, 40, 50]` and `[40, 50, 10, 20, 30]` are *circularly identical*, but not with `[50, 40, 10, 20, 30]` .

Write a function which decides whether the given 2 parameter lists are circularly identical or not and returns a boolean value accordingly (*True* or *False*).

In [18]:

```python
def is_circularly_identical(listA, listB):
    listA2 = listA + listA
    for i in range(len(listA2)):
        if listA2[i] == listB[0]:
            n = 1
            while (n < len(listB)) and (i + n < len(listA2)) and (listA2[i+n] ==
 listB[n]):
                n += 1
            if n == len(listB):
                return True

    return False



list1 = [10, 20, 30, 40, 50]
list2 = [40, 50, 10, 20, 30]
list3 = [50, 40, 10, 20, 30]

print('list1 is circularly identical with list2: {0}'.format(is_circularly_ident
ical(list1, list2)))
print('list1 is circularly identical with list3: {0}'.format(is_circularly_ident
ical(list1, list3)))
print('list2 is circularly identical with list3: {0}'.format(is_circularly_ident
ical(list2, list3)))
```

```
list1 is circularly identical with list2: True
list1 is circularly identical with list3: False
list2 is circularly identical with list3: False
```