Introduction to programming

Labor05

Revision

- Input numbers/Read values
  - Until 0
  - Until a negative number
  - Until EOF – End-Of-File (CTRL+D)
- Exceptions
  - try – except
  - try – finally
- Strings
  - String functions
  - Slicing strings

Homework 1

- Write a program that determines and prints to the standard output **how many positive even values** were present at the standard input.
- The first line of the input contains a positive integer (n), which is the number of rows containing integers to be processed. Each of the next n rows contains exactly one integer.
- The program should write exactly one integer to the output, which is satisfy the condition.
- **Input:**
  - 5
  - -2
  - -1
  - 0
  - 1
  - 2
- **Output:**
  - 1

Solution

```
n = int(input())
c = 0

for i in range(n):
    num = int(input())
    if num > 0 and num % 2 == 0:
        c += 1

print(c)
```

Homework 2

- Write a program that reads words up to (end–of–file) *EOF* and removes/deletes the vowels from the input words.

- Input:
  - apple
  - pear
  - peach

- Output:
  - ppl
  - pr
  - pch

Solution – 1

```
import sys

for s in sys.stdin:
    r = ''
    for c in s.strip():
        if c.lower() not in "aeiou":
            r = r + c
    print(r)
```

Solution – 2

```
import sys

for s in sys.stdin:
    r=''.join([c for c in s if c.lower() not in "aeiou"])
    print(r)
```

Solution – 3

```
while True:
    try:
        s = input()
        r = ''
        for c in s:
            if c.lower() not in "aeiou":
                r = r + c
        print(r)
    except EOFError:
        break
```

Solution – 4

```
while True:
    try:
        s = input()
        r=''.join([c for c in s if c.lower() not in "aeiou"])
        print(r)
    except EOFError:
        break
```

Solution – 5

```
import sys

for s in sys.stdin:

    s = s.strip()
    i = 0
    while i < len(s):
        if s[i].lower() in 'aeiou':
            s = s.replace(s[i], '')
        i = i + 1
    print(s)
```

Homework 3

- Write a program that reads words up to the *"END"* string, and doubles/duplicates the uppercase letters.

- Input:
  - Apple
  - PeaR
  - PeaCh
  - END

- Output:
  - AApple
  - PPeaRR
  - PPeaCCh

Solution – 1

```
s = input()
while s != 'END':
    r = ''
    for c in s:
        if c.isupper():
            r += c * 2
        else:
            r += c
    print(r)
    s = input()
```

Solution – 2

```python
while True:
    s = input()
    if s == 'END':
        break
    r = ''
    for c in s:
        if c.isupper():
            r += c * 2
        else:
            r += c
    print(r)
```

Solution – 3

```
s = input()
while s != 'END':
    r = ''.join([c * 2 if c.isupper() else c for c in s])
    print(r)
    s = input()
```

Exercise – GCD
https://progcont.hu/progcont/b50/?pid=200311

- Write a program that reads two positive integers from the standard input and determines their greatest common divisor.

Input Specification

- The input contains two positive integers separated by a space character.

Output Specification

- The program should write a single line to the standard output containing the greatest common divisor of the two input number.

Sample Input

- 15 10

Output for Sample Input

- 5

Solution 1 – Euclidean Algorithm

```
line = input()
s = line.split(" ")
a = int(s[0])
b = int(s[1])

while a % b != 0:
    r = a % b
    a = b
    b = r

print(b)
```

Solution 1 – Euclidean Algorithm

```
line = input()
s = line.split(" ")
a = int(s[0])
b = int(s[1])

while a % b:
    r = a % b
    a = b
    b = r

print(b)
```

Solution 2

```
line = input()
s = line.split(" ")
a = int(s[0])
b = int(s[1])

while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a

print(b)
```

Solution 3

```
line = input()
s = line.split(" ")
a = int(s[0])
b = int(s[1])
gcd = 1
if a < b:
    c = a
else:
    c = b
for i in range(2, c + 1):
    if (a % i == 0 and b % i == 0):
        gcd = i

print(gcd)
```

Least Common Multiple

- The least common multiple, lowest common multiple, or smallest common multiple of two integers a and b, is the smallest positive integer that is divisible by both a and b.
- $LCM = \frac{a \cdot b}{GCD}$

Exercise – Relative Primes https://progcont.hu/progcont/b50/?pid=200312

- Write a program that reads two positive integers from the standard input and decides whether they are relative primes or not.

Input Specification

- The input contains two positive integers separated by a space character.

Output Specification

- The program should write a single line to the standard output containing the string „IGEN” („yes” in Hungarian) or „NEM” („NO” in Hungarian) depending on whether the two input numbers are relative primes.
- **Sample Input**
- 15 10

Output for Sample Input

- NEM

Solution

```python
line = input()
s = line.split(" ")
a = int(s[0])
b = int(s[1])
while a != b:
    if a > b:
        a = a - b
    else:
        b = b - a

if a == 1:
    print("IGEN") #yes
else:
    print("NEM") #no
```

Random numbers

- we often use random numbers in mathematical and statistical problems
- Python includes a random number generator
- **`import`** `random` – import module
- Random module functions:
  - `random()` – random number between [0.0, 1.0)
  - `uniform(a,b)` – random number within the interval [a,b]
  - `randint(a,b)` – integer random number within interval [a,b]
  - `choice()` – returns a randomly selected element from the specified sequence.

Exercise

- Write a program that calculates the product of n randomly generated integers between [1,10].

Solution

```python
import random

n = int(input("n = "))
p = 1
for i in range(1, n + 1):
    x = random.randint(1, 10)
    print(x, end=' ')
    p = p * x
print("Product = ", p)
```

Exercise

- Write a program that generates numbers between 1 and 10, and exits when we input 5.
- Print out the number of steps.

Solution

```
import random

c = 0
while True:
    n = random.randint(1, 10)
    if n == 5:
        break
    print(n, end=' ')
    c += 1
print("Stopped after {} steps!".format(c))
```

Lists

- A list is a collection of values, in fact a container that can contain numbers, texts, etc.
- The values that make up a list are called elements.
- The elements are numbered, **indexable.**
- The elements of a list can be of any type.
- Create an empty list:

```
    list = []
```

Lists

- Length of the list: `len()`
- List traversal: `a = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]`

```
for i in a:
   print(i)

for i in range(len(a)):
    print(a[i])

i = 0
while i < len(a):
    print(a[i])
    i += 1
```

List operations

- + operator concatenates the lists
- * operator repeats the list the specified number of times
- slicing operator: remove items from a list by assigning an empty list to a slice of the list, or add an item to a list by inserting an empty slice at the desired location.
- del() – deletes the given list item

Lists

```
>>> [1,2,3]
[1, 2, 3]
>>> a = [1,2,3]
>>> list = [] #empty list
>>> list
[]
>>> a = [1,2,'ab',3.14]
>>> a
[1, 2, 'ab', 3.14]
>>> len(a) #length of the list
4
>>> [1,2]+[3,4] #concatenate
[1, 2, 3, 4]
```

Lists

```
>>> a=[1,2,3]
>>> b = a # refer to a reference
>>> b
[1, 2, 3]
>>> a[0]
1
>>> a[0]=10
>>> a
[10, 2, 3]
>>> b
[10, 2, 3]
>>> b=a[:] # a full copy of a
>>> b
[10, 2, 3]
>>> a[0]=20
>>> b
[10, 2, 3]
>>> a
[20, 2, 3]
```

Slicing lists

```
>>> a
[20, 2, 3]
>>> a == b
False
>>> a[1:]
[2, 3]
>>> a[-1]
3
>>> a[:1]
[20]
>>> a*3
[20, 2, 3, 20, 2, 3, 20, 2, 3]
```

List operations

```
>>> a=[1,2,3]
>>> a.append(20) #append a new element to the end of the list
>>> a
[1, 2, 3, 20]
>>> a.pop(0) # removes the element and returning its value
1
>>> a
[2, 3, 20]
>>> del a[2]
>>> a
[2, 3]
```

List operations

```
>>> a=[1,2,3,4,5,6,7,8]
>>> a[2:4]
[3, 4]
>>> a[2:4]=[] # replacing more elements
>>> a
[1, 2, 5, 6, 7, 8]
>>> a[2:4]=[3,1,2] # inserting more elements
>>> a
[1, 2, 3, 1, 2, 7, 8]
```

Sorting the list

```
>>> a=[5, 2, 4, 8, 3, 1]
>>> a
[5, 2, 4, 8, 3, 1]
>>> sorted(a)
[1, 2, 3, 4, 5, 8]
>>> sorted(a, reverse=True)
[8, 5, 4, 3, 2, 1]
>>> a
[5, 2, 4, 8, 3, 1]
>>> a = sorted(a)
>>> a = ['c','b','a','d']
>>> a.sort()
>>> a
['a', 'b', 'c', 'd']
```

List operations

```
>>> list = [9, 3, 4, 6, 8, 5, 3, 1, 5]
>>> max(list)
9
>>> min(list)
1
>>> sum(list)
44
>>> list.count(3)
2
```

List operations

- `append(element)` – appends the element specified as argument to the end of the list. The list must already exist, e.g. an empty list.
- `insert(position, element)` – inserts the element given as argument into the list at the given position, in which case the rest of the list moves one position to the right of the given position.
- `extend(list)` – appends all elements of the list specified as argument to the end of the list.
- `clear()` – clears all elements of the list (does not clear the list itself).

List operations

- `remove(element)` – deletes the first occurrence of the item specified as argument from the list.
  If the argument does not occur in the list, the program terminates with a runtime error.
- `pop(index)` – removes the element with the position specified as argument from the list, returning its value.
- `count(element)` – counts the number of times the element specified as argument is in the list.

List operations

- `index(elem)` return the index of the first occurrence of the element in the list given as an argument. If the element given as argument is not in the list, it is stopped with an error message.
- `reverse()` – reverses the elements in the list.
- `sort()` – sorts the elements in the list.
  If the list elements are numbers, they are in ascending order, if they are strings, they are in alphanumeric order.
  If the elements of the list are of mixed type, it terminates with a runtime error.

List Comprehension

The Syntax:

- newlist = [*expression* for *item* in *iterable* if *condition* == True]

```
>>> list = [x for x in range(10)]
>>> list
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

>>> newlist = [x for x in list if x % 2 == 0]
>>> newlist
[0, 2, 4, 6, 8]
```

List Comprehension

```
fruits =
["apple", "banana", "cherry", "kiwi", "mango"]
newlist = []

for x in fruits:
  if "a" in x:
    newlist.append(x)

#newlist = [x for x in fruits if "a" in x]

print(newlist)
```

Examples

```
fruits = ["apple", "banana", "cherry",
"kiwi", "mango"]

newlist = [x.upper() for x in fruits]

print(newlist)
```

Exercise – Practice

- The following tasks should be solved using "list comprehension".

Task1

- ['anthony', 'blake', 'charlotte'] → ['Anthony', 'Blake', 'Charlotte']

Task2

- [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], i.e. initialize a 10-item list with all 0's.

Task3

- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] → [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

Task4

- ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'] → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] (first list contains strings)

Exercise – Practice

- The following tasks should be solved using "list comprehension".

Task5

- "1234567" → [1, 2, 3, 4, 5, 6, 7] i.e. we have numbers in string format and we want to put the digits (as numbers) in a list

Task6

- 'The quick brown fox jumps over the lazy dog' → [3, 5, 5, 3, 5, 4, 3, 4, 3], i.e. find the length of each word

Task7

- "python is an awesome language" → ['p', 'i', 'a', 'a', 'l'], i.e. collect the initial letters of the words in a string in a list

Solutions

Task1

- ['anthony', 'blake', 'charlotte'] → ['Anthony', 'Blake', 'Charlotte']

```
#Solution 1
l = ['anthony', 'blake', 'charlotte']
new = [x[0].upper() + x[1:] for x in l]
print(new)

#Solution 2
l = ['anthony', 'blake', 'charlotte']
new = [x.capitalize() for x in l]
print(new)
```

Solutions

Task2

- [0, 0, 0, 0, 0, 0, 0, 0, 0, 0], i.e. initialize a 10-item list with all 0's.

```
new = [0 for i in range(10)]
print(new)
```

Task3

- [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] → [2, 4, 6, 8, 10, 12, 14, 16, 18, 20]

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
new = [2 * i for i in l]
print(new)
```

Solutions

Task4

- ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10'] → [1, 2, 3, 4, 5, 6, 7, 8, 9, 10] (first list contains strings)

```
l = ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
new = [int(i) for i in l]
print(new)
```

Task5

- "1234567" → [1, 2, 3, 4, 5, 6, 7] i.e. we have numbers in string format and we want to put the digits (as numbers) in a list

```
s = "1234567"
new = [int(c) for c in s]
print(new)
```

Solutions

Task6

- 'The quick brown fox jumps over the lazy dog' → [3, 5, 5, 3, 5, 4, 3, 4, 3], i.e. find the length of each word

```
s = 'The quick brown fox jumps over the lazy dog'
new = [len(c) for c in s.split()]
print(new)
```

Task7

- "python is an awesome language" → ['p', 'i', 'a', 'a', 'l'], i.e. collect the initial letters of the words in a string in a list

```
s = 'python is an awesome language'
new = [c[0] for c in s.split()]
print(new)
```

Exercise

- Fill a list with n **even numbers** generated by a random number generator between 1 and 10.

Solution

```python
import random

n = int(input("n="))
list = []
while True:
    if len(list) == n:
        break
    x = random.randint(1, 10)
    if x % 2 == 0:
        list.append(x)
print(list)
```

Exercise

- Fill up a list of n elements with integers between [1, 10], and print out:
  - the elements of the list
  - the sum of the elements of the list
  - the minimum and the maximum elements
  - check that all the elements of the list are even
  - sort the even elements in ascending order
  - sort the odd elements in descending order

```python
import random

list = []
evens = True
even = []
odd = []
n = int(input('n='))
for i in range(n):
    list.append(random.randint(1, 10))
    if list[i] % 2 != 0:
        odd.append(list[i])
        evens = False
    else:
        even.append(list[i])

print(list)
print("Sum =", sum(list))
print("Evens =", evens)
print("Min =", min(list))
print("Max =", max(list))

even = sorted(even)
odd = sorted(odd, reverse=True)

print("Evens =", even)
print("Odds =", odd)
```

Homework 1– Neandervölgyi gén https://progcont.hu/progcont/100350/?pid=201515

Neanderthal gene

- A recent study suggests that the risk of coronavirus infection is more severe in people who carry a small Neanderthal gene. Reading this makes you stop for a moment and wonder if you have even an ounce of the Neanderthal gene in you.
- If you make a model, Neanderthal genes are relatively easy to identify. Suppose we describe a DNA molecule by a sequence of integers. If we can find at least three consecutive values in this sequence that form a strictly monotonic series, we can say that the DNA molecule contains a Neanderthal gene.
- Write a program that reads at least one positive integer per line from the standard input to the end of file (EOF). Each line describes a DNA molecule. If the DNA molecule under test contains a Neanderthal gene, your program should write a line containing a "YES" string to the standard output, otherwise a "NO" string.

Homework 1– Neandervölgyi gén
https://progcont.hu/progcont/100350/?pid=201515

Sample Input
```
5 6 7
9 8 7 6 5
2 3
7 11 8 10 9
4 4 4 4
9
1 2 3 1 2
7 8 7 8 7 8 9
```
Output for Sample Input
```
YES
YES
NO
NO
NO
NO
YES
YES
```

Homework 2 – Malacpersely https://progcont.hu/progcont/100353/?pid=201527

- **Piggy bank**
- Children love to collect their savings in piggy banks. In this exercise, you have to make a statement of these savings.
- Write a program that reads the names of the children from standard input to end-of-file (EOF) and the money they have saved from week to week, line by line!
- The form of a line is:
- `name:quantity[ quantity]...`
- Your program will write in the first line of the standard output how much money the children have saved in total according to the statement! While processing the data, also count the number of children who regularly saved at least 20 forints in their piggy bank every time! Print the latter value on the second line of the standard output!

Homework 2 – Malacpersely
https://progcont.hu/progcont/100353/?pid=201527

Sample Input

- Mark:20 30 40 50 60
- Jogh:25 15 25 15 25 15
- Peter:30
- David:20 30 10 30 20 10 20 30
- Julie:19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1

Output for Sample Input

- 710
- 2

Homework 3 – Fordított számsorok
https://progcont.hu/progcont/100348/?pid=201508

- **Reverse numbers**
- Write a program that reads a sequence of numbers from the standard input and writes its elements in reverse order to the standard output.
- Each line of the input contains at least one integer, and the consecutive numbers are separated by exactly one space.
  The first number (m) in each line is the number of elements in the number line to be processed.
  The program must write the following sequence of m integers in reverse order to the standard output. The end of the input is indicated by a line containing only one 0.
  he program need not process this line.
- Make sure that the numbers written in a line are separated by exactly one space, and do not print spaces at the beginning and end of the line.

Homework 3 – Fordított számsorok

https://progcont.hu/progcont/100348/?pid=201508

Sample Input

```
4 1 2 3 4
5 2 5 3 4 1
2 2 3
0
```

Output for Sample Input

```
4 3 2 1
1 4 3 5 2
3 2
```