# Introduction to programming

Labor03

## Revision

- `format()` function
- Number systems – conversions
  - p->10
  - 10->p
  - Connection between 2<->8 and 2<->16
- Basic algorithm structures
- Selection
  - if-else statement

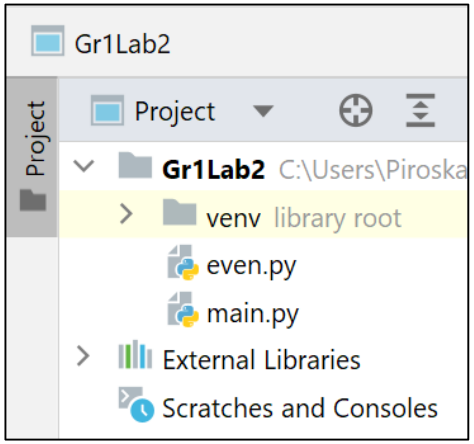
## The Pass statement – Revision

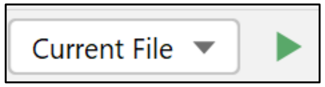
- It has a placeholder role
- Do nothing
- Empty statement

```
n = int(input("n = "))
if n <= 100:
    pass
else:
    print("Wrong value!")
```

# PyCharm – Community Edition!!!

- **New Project**
  - **GrXLabX**
  - Pl. Gr1Lab3 <- Group1 Lab3
- **New Python file**
  - filename.py
- **Run**
  - Current file / Edit configuration -> filename.py
  - Green icon
  - Shift+F10
  - Menu: Run->Run filename.py





## Exercise – Homework 1

- **Make a simple calculator:** Write a program which inputs two integer numbers and an operation symbol (+,-,*,/, //, %).
- On the basis of the operation symbol calculate the result.
- Give an error message if the operation is not what we have listed or in case of the division the denominator is zero!
- **Input:** a, b, op
- **Output:** a op b = result
- **For example:**
  - 5 + 2 = 7
  - 5 - 2 = 3
  - 5 * 2 =10
  - 5 / 2 = 2.5
  - 5 // 2 = 2
  - 5 % 2 = 1

```python
op = input("""Choose from the following operators:
+ addition
- substraction
* multiplication
/ division
// float division
% remainder
op = """)
a = int(input("a = "))
b = int(input("b = "))
if op == '+':
    print("{} + {} = {}".format(a, b, a + b))
elif op == '-':
    print("{} - {} = {}".format(a, b, a - b))
elif op == '*':
    print("{} * {} = {}".format(a, b, a * b))
elif (op == '/' or op == '//' or op == '%') and b == 0:
    print("Error! Division by zero!")
elif op == '/':
    print("{} / {} = {:.2f}".format(a, b, a / b))
elif op == '//':
    print("{} // {} = {}".format(a, b, a // b))
elif op == '%':
    print("{} % {} = {}".format(a, b, a % b))
else:
    print("Wrong operator!")
```

## Exercise – Homework 2

- Write a program which reads the coefficients of a second-degree equation and then print out the roots of the equation.

- **Input**: a, b, c
- **Output**: $x_1$, $x_2$

$$ax^2 + bx + c = 0$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

# Solution

```python
import math

print("Input the coefficients of the
second-degree equation!")
a = int(input("a="))
b = int(input("b="))
c = int(input("c="))
if a == 0:
    if b == 0:
        print("Error!")
    else:
        x = -c / b
        print("First-degree, x=", x)
else:
     d = b * b - 4 * a * c
     print("The value of the
     discriminant:", d)
```

```python
if d > 0:
    print("Two real solutions.")
    x1 = (-b - math.sqrt(d)) / (2 * a)
    x2 = (-b + math.sqrt(d)) / (2 * a)
    print("x1 =", x1)
    print("x2 =", x2)
elif d == 0:
    x = -b / (2 * a)
    print("One solution: ", x)
else:
    print("Not real solution!")
```

# Exercise – Homework 3

- Write a program that prints the number of days in a month according to a date that has been read.

- **Input:**
  - Year
  - Month

- **Output**:
  - Day
    - 2023, 1 -> **31**
    - 2023, 4 -> **30**
    - 2023, 2 -> **28**
    - 2000, 2 -> **29 #leap year**

# Leap Year

- A **leap year** is a calendar year that contains an additional day added to keep the calendar year synchronized with the astronomical year or seasonal year.
- The extra day is added in **February**, making it **29 days** instead of the normal 28 days.
- Leap years occur every 4 years. 2024 is a leap year and so is 2028, 2032 and so on.

- **How to know if it is a Leap Year**:
  - Leap Years are any year that can be **exactly divided by 4** (such as 2016, 2020, 2024, etc)
  - *except if* it can be **exactly divided by 100**, then it **isn't** (such as 2100, 2200, etc)
  - *except if* it can be **exactly divided by 400**, then it **is** (such as 2000, 2400)

## Solution

```
year = int(input("Year = "))
month = int(input("Month = "))

if month == 1 or month == 3 or month == 5 or month == 7 or
   month == 8 or month == 10 or month == 12:
    day = 31
elif month == 4 or month == 6 or month == 9 or month == 11:
    day = 30
elif month == 2:
    if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
        day = 29
    else:
        day = 28
else:
    print("Wrong value!")
    day = 0
print("Number of days = ", day)
```

# Match statement – Python 3.10

```
match value:         #parameter
    case value1:     #pattern
        statement1
    case value2:
        statement2
    case value3:
        statement3
    ...
    case valuen:
        statementn

    case _:          #else
        statement
```

## Exercise

- Write a program that evaluates the input value:
  - 1 – Fail
  - 2 – Pass
  - 3 – Satisfactory
  - 4 – Good
  - 5 – Excellent

Use the `match` statement!

## Match statement

```
point = int(input("point="))
match point:
    case 1:
        print("Fail!")
    case 2:
        print("Pass!")
    case 3:
        print("Satisfactory!")
    case 4:
        print("Good!")
    case 5:
        print("Excellent!")
    case _:
        print("Wrong value!")
```

## Exercise

- Write a program that prints the number of days in a month according to a date that has been read.

- **Input:**
  - Year
  - Month

- **Output**:
  - Day
    - 2023, 1-> **31**
    - 2023, 4 -> **30**
    - 2023, 2 -> **28**
    - 2000, 2 -> **29 #leap year**

# Solution

```
year = int(input("Year = "))
month = int(input("Month = "))
match month:
    case 1 | 3 | 5 | 7 | 8 | 10 | 12:
        day = 31
    case 4 | 6 | 9 | 11:
        day = 30
    case 2:
        if (year % 4 == 0 and year % 100 != 0) or year % 400 == 0:
            day = 29
        else:
            day = 28
     case _:
         print("Wrong value!")
         day = 0
print("Number of days = ", day)
```

## Loops

- Loops are program control structures that allow you to execute one or more statements more than once.

**Characteristics of loops**:

- They contain a block of statements called a **loop core**: a loop core can be executed more than once.
- They contain a **repetition condition** (control condition) that determines whether the loop core must be repeated.
- Contain(s) an instruction to terminate the loop.
- Contain an initial value (before the loop) to ensure that the loop condition can be evaluated unambiguously the first time.

## Loops

Python has two primitive loop commands:

- for,
- while.

## The for loop

A **for loop** is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

The syntax of the for loop:

```
for loopvariable in sequence:
     statements
```

Also known as:

- Counted loops

## The for loop

- If the sequence contains a list of terms, this is evaluated first.
- Then the first element of the sequence is a loop variable.
- The next step is to execute the loop core and then continue with the second element of the sequence.
- The loop core is repeated until the sequence is finished.

## The range() function

- **range**(start value, end value, step value)

start value = 0

- `range(6) #range(0,6,1)`

step value = 1

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

Attention!!!
end value−1

- `range(5, 10)`

```
5, 6, 7, 8, 9
```

- `range(0, 10, 3)`

```
0, 3, 6, 9
```

- `range(-10, -100, -30)`

```
-10, -40, -70
```

- `range(10, 0, -1) #backwards`

```
10, 9, 8, 7, 6, 5, 4, 3, 2, 1
```

## Exercise

- Write a program that prints out the integer numbers from 10 to 20.

## Solution

```
for i in range(10, 21):
    print(i)
```

## Exercise

- Write a program that prints out the integer numbers and their squares up to an input value.

## Solution

```
n = int(input("n ="))
for i in range(1, n + 1):
    print("{0:2d} {1:4d}".format(i, i * i))
```

## Exercise

- Write a program that prints out the numbers divisible by 4 from 1 up to an input number.

## Solution – 1

```
n = int(input("n = "))
for i in range(4, n + 1, 4):
    print(i, end=" ")
```

## Solution – 2

```
n = int(input("n = "))
for i in range(4, n + 1):
    if (i % 4 == 0):
        print(i, end=" ")
```

## Exercise

- Write a program that prints the greatest real divisor of an input number.

## Solution

```
n = int(input("n = "))
div = 1
for i in range(2, n):
    if (n % i == 0):
        div = i

if div == 1:
    print("Prime number!")
else:
    print("Greatest real divisor: ", div)
```

## Exercise

- Write a program that calculates the factorial of an input number.

## Solution – 1

```
n = int(input("n = "))
f = 1
for i in range(1, n+1):
    f *= i

print("{0}! = {1}".format(n, f))
```

## Solution – 2

```
n = int(input("n = "))
f = 1
for i in range(n, 0, -1):
    f *= i

print("{0}! = {1}".format(n, f))
```

## Exercise

- Write a program that calculates the sum of the numbers which are divisible by 3 between two input numbers.

## Solution

```python
a = int(input("a = "))
b = int(input("b = "))
s = 0
for i in range(a, b + 1):
    if i % 3 == 0:
        s += i

print("Sum = ", s)
```

## Exercise

- Write a program that gets 5 integer numbers as inputs and calculates their average.

## Solution

```
s = 0
for i in range(1, 6):
    n = int(input("n = "))
    s += n

print("Average = {0:.2f}".format(s / 5))
```

## The while loop

- With the while loop we can execute a set of statements as long as a condition is true.
- The syntax of the while loop:

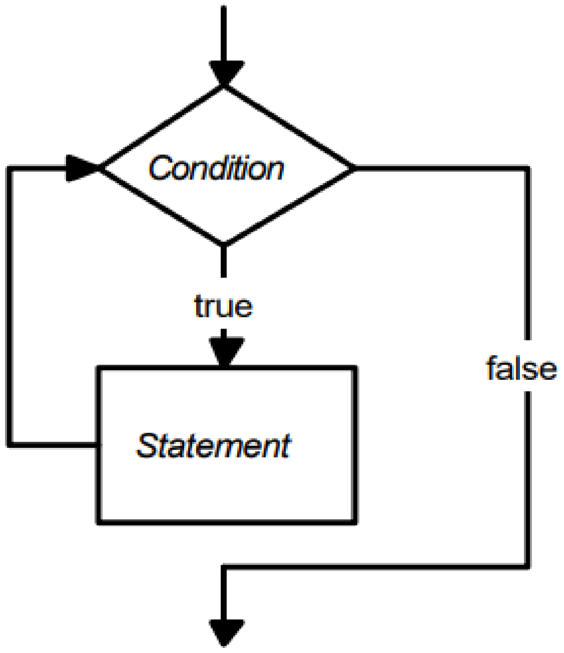
```
while expression:
    statements
```

- starts with the **while** keyword
- the expression must be a **logical expression** as a condition for remaining in the loop
- the statement is the **core of the loop**

# The while loop

- Evaluation:

- 1. evaluates the expression (condition):
  - if FALSE, exits the loop,
  - if TRUE, then:
- 2. executes the loop core
- 3. returns to step 1



## The while loop

- it is useful to use the while loop if you cannot predefine the number of iterations (it is decided at runtime), or if the stop depends on the result of some expression that provides a logical value

- **pretest loop**, checks the loop condition before the loop core is executed, if the condition is false, the loop core does not run at all

## Exercise

- Write a program that prints out the integer numbers from 1 to 10 and their squares on the standard output, one under the other.

## Solution

```
i = 1
# declaration of the loop variable

while i <= 10:
# loop condition: run until i is less than equal
to 10

     print("{0:2d} {1:4d}".format(i, i*i))
     i += 1
     # increase the loop variable
```

## Exercise

- Calculate the sum of the numbers from 1 to an input integer number (N).

## Solution

```python
n=int(input("n = "))
s = 0
i = 1
while i <= n:
    s = s + i
    i = i + 1
print("Sum = ", s)
```

## Exercise

- Calculate the product of the numbers from 1 to an input integer number (N).

## Solution

```python
n=int(input("n = "))
p = 1
i = 1
while i <= n:
    p = p * i
    i = i + 1

print("Product = ", p)
```

## Exercise

- Write a program that calculates the numbers of the digits of the input number.

## Solution

```
n = int(input("n = "))

counter = 0
while n != 0:
      counter = counter + 1
      n = n // 10
print("Digits= ", counter)
```

## Exercise

- Write a program that calculates the sum of the digits of the input number.

## Solution

```
n = int(input("n = "))

sum = 0
while n != 0:
      sum += n % 10
      n = n // 10
print("Sum of digits= ", sum)
```

## Exercise

- Write a program that prints out the prime factor of the input number.

## Solution

```
n = int(input("n = "))
i = 2
while n > 1:
    if n % i == 0:
        n /= i
        if n != 1:
            print(i, end=" ")
        else:
            print(i)
    else:
        i += 1
```

## Exercise

- Write a program that reads integer numbers until 0 and prints out the minimum and maximum of the input numbers.

## Solution

```
n = int(input("n = "))
min = n
max = n
while n:    #while n!=0:
    if n < min:
        min = n
    if n > max:
        max = n
    n = int(input("n = "))
print("Min = ", min)
print("Max = ", max)
```

## Homework 1

- **Maximum of Integer Numbers #1**
- https://progcont.hu/progcont/b50/?pid=200309

## Homework 2

- **Grading #3**
- https://progcont.hu/progcont/b50/?pid=200316

## Homework 3

- **Average of Integers #1**
  https://progcont.hu/progcont/b50/?pid=200304
- **Average of Integers #3**
  https://progcont.hu/progcont/b50/?pid=200306