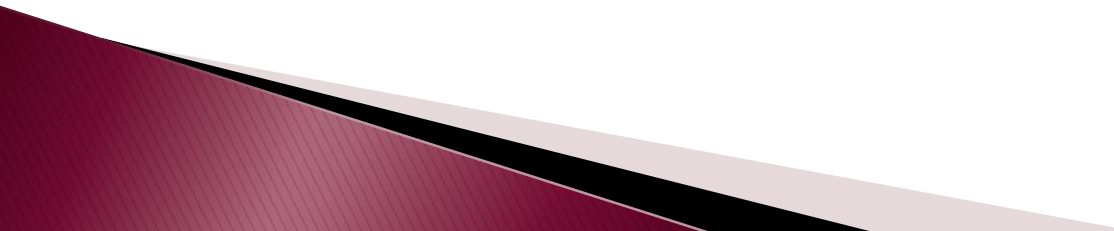


# Introduction to programming

Labor02

# Revision

- ▶ Variables
  - ▶ Types
  - ▶ Type conversion
  - ▶ Operators
  - ▶ Input
  - ▶ Output
  - ▶ Math module
- 
- A decorative gradient bar at the bottom of the slide, transitioning from dark red on the left to light grey on the right.

# Homework 1

- ▶ Installation
- ▶ Math module

# Homework 2

- ▶ Input the radius of a circle, the program have to calculate its circumference and area (use the PI constant defined in the **math** module: `math.pi`).

# Solution

```
import math

r = int(input("Radius:"))
a = math.pi * r ** 2
c = 2 * math.pi * r
print("Area:", a)
print("Circumference:", c)
```

# Homework 3

- ▶ Input two integers, write a program which calculate the arithmetic and geometric mean of the numbers.

# Solution

```
import math

a = int(input("a = "))
b = int(input("b = "))

print("Arithmetic mean: ", (a+b)/2)
print("Geometric mean: ", math.sqrt(a*b))
```

# Format function

- > `print("Pi value = {:.2f}".format(math.pi))`
- > `print("Pi value = {0:.2f}".format(math.pi))`
- > `print("Pi value = {pi:.2f}".format(pi=math.pi))`
- > `print("Name={}, price={}".format("apple", 350))`
- > `print("Name={0}, price={1}".format("apple", 350))`
- > `print("Name={name}, price={price}".format(name="apple", price=350))`



# Number systems

- ▶ Decimal number system
  - the concept of place value
- ▶ Generally the numbers can be described in the following form:

$$a_n a_{n-1} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-m}$$
$$\sum_{i=-m}^n a_i \cdot 10^i, \text{ when } 0 \leq a_i < 10.$$

- ▶ Optional in the  $p$  base ( $p > 1$ ) number system the used digits 0, 1, ...,  $p-1$ , the place values are the powers of number  $p$ :

$$\sum_{i=-m}^n a_i \cdot p^i, \text{ when } 0 \leq a_i < p.$$

**Example:** 123,45

# Number systems

- ▶ Binary (base two) number system
  - digits: 0, 1
- ▶ Ternary (base three) number system
  - digits: 0, 1, 2
- ...
- ▶ Octonary (base eight) number system
  - digits: 0, 1, 2,..., 7
- ...
- ▶ Decimal (base ten) number system/ten base
  - digits: 0, 1, 2,..., 9
- ...
- ▶ Hexadecimal (base sixteen) number system
  - digits: 0, 1, 2, ..., 9, A, B, C, D, E, F

Binary p = 2	Ternary p = 3	Quinary p = 5	Octonary p = 8	Decimal p = 10	Duodecimal p = 12	Hexadecimal p = 16
0	0	0	0	0	0	0
1	1	1	1	1	1	1
10	2	2	2	2	2	2
11	10	3	3	3	3	3
100	11	4	4	4	4	4
101	12	10	5	5	5	5
110	20	11	6	6	6	6
111	21	12	7	7	7	7
1000	22	13	10	8	8	8
1001	100	14	11	9	9	9
1010	101	20	12	10	a	A
1011	102	21	13	11	b	B
1100	110	22	14	12	10	C
1101	111	23	15	13	11	D
1110	112	24	16	14	12	E
1111	120	25	17	15	13	F
10000	121	26	20	16	14	10

# Number systems base names

- 2 binary
- 3 ternary
- 4 quaternary
- 5 quinary
- 6 senary
- 7 septenary
- 8 octonary
- 9 nonary
- 10 decimal
- 11 undenary
- 12 duodecimal
- 13 tridecimal
- 14 quattuordecimal
- 15 quindecimal
- 16 hexadecimal

# Convert to base 10 (p→10)

Generally:

$$\begin{aligned} & a_n a_{n-1} \dots a_0 . a_{-1} a_{-2} \dots a_{-m} \text{ }_p = \\ & a_n \cdot p^n + a_{n-1} \cdot p^{n-1} + \dots + a_0 \cdot p^0 + a_{-1} \cdot p^{-1} + \\ & \quad + a_{-2} \cdot p^{-2} + \dots + a_{-m} \cdot p^{-m} \\ & 0 \leq a_i < p \end{aligned}$$

For example:

$$\begin{aligned} 263.15_7 &= 2 \cdot 7^2 + 6 \cdot 7^1 + 3 \cdot 7^0 + 1 \cdot 7^{-1} + 5 \cdot 7^{-2} = \\ &= 98 + 49 + 3 + \frac{1}{7} + \frac{5}{7^2} = 143 \frac{12}{49}_{10} \end{aligned}$$

# Exercise

- Convert each of the following different base representation to its equivalent **base ten** form:

1.  $1011100.101_2 =$

2.  $1221.12_3 =$

3.  $152.43_6 =$

4.  $173.104_8 =$

5.  $841.47_9 =$

6.  $13A.2F_{16} =$

# Convert from base 10 to different base number representations (10→p)

113.45<sub>(10)</sub>

113	2
56	1
28	0
14	0
7	0
3	1
1	1
0	1



0	.45	2
0	.90	
1	.8	
1	.6	
1	.2	
0	.4	
0	.8	
1	.6	
1	.2	
0	.4	
0	.8	

1100001.0111001100<sub>(2)</sub>

# Exercise – Homework

- ▶ Convert each of the following base ten representation to its equivalent **binary** form:

1.  $262_{10} =$

2.  $32.3_{10} =$

3.  $171.64_{10} =$

- ▶ Convert each of the following ten base representation to its equivalent **octonary** form:

1.  $51_{10} =$

2.  $71.8_{10} =$

3.  $117.18_{10} =$

- ▶ Convert each of the following ten base representation to its equivalent **hexadecimal** form:

1.  $33_{10} =$

2.  $81.9_{10} =$

3.  $218.12_{10} =$



# Connection between the octal and the binary system

Binary	Octal
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

# Connection between the hexadecimal and the binary system

Binary	Hexadecimal	Binary	Hexadecimal
0000	0	1000	8
0001	1	1001	9
0010	2	1010	A
0011	3	1011	B
0100	4	1100	C
0101	5	1101	D
0110	6	1110	E
0111	7	1111	F

# Exercise – Homework

- ▶ Convert each of the following binary representation to its equivalent octal and hexadecimal form, and each of the following hexadecimal representation to its equivalent binary and octal form:

1.  $1110\ 1001\ 1100\ 0011_2 =$

2.  $1011\ 0111\ 0101\ 0100_2 =$

3.  $1000\ 1101\ 1111\ 0011\ 1101_2 =$

4.  $1010\ 1011\ 0011\ 1110\ 0001\ 0101_2 =$

5.  $3BCF_{16} =$

6.  $BF29_{16} =$

7.  $48C5_{16} =$

8.  $63AE_{16} =$

# Webpages – Practicing – Converters

- ▶ <http://planetcalc.com/862/>
- ▶ [http://courses.cs.vt.edu/~cs1104/number\\_conversion/conv2.html](http://courses.cs.vt.edu/~cs1104/number_conversion/conv2.html)

# Format function

```
> print("Binary {0:b} decimal = {0:d}" .format(0b101))
> print("Octal {0:o} decimal = {0:d}" .format(0o17))
> print("Hexadecimal {0:x} decimal = {0:d}" .format(0xab))
> print("Hexadecimal {0:X} decimal = {0:d}" .format(0XB2F))

> print("{0} binary = {0:b}".format(5))
> print("{0} binary = {0:b}".format(123))
> print("{0} octal = {0:o}".format(12))
> print("{0} hexadecimal = {0:x}".format(12))
> print("{0} hexadecimal = {0:X}".format(12))
```

# Functions

- ▶ **bin()** – is an in-built function that takes in integer x and returns the binary representation of x in a string format.
  - `>>> bin(15)`
  - `'0b1111'`
- ▶ **oct()** – is an in-built function that takes in integer x and returns the octonary representation of x in a string format.
  - `>>> oct(15)`
  - `'0o17'`
- ▶ **hex()** – is an in-built function that takes in integer x and returns the hexadecimal representation of x in a string format.
  - `>>> hex(15)`
  - `'0xf'`

# Bitwise operators

Operator	Description
&	Bitwise and
	Bitwise or
^	Bitwise xor
~	Bitwise not
<<	Bitwise left shift
>>	Bitwise right shift

# Logical operators

Operator	Description
and	Logical and
or	Logical or
not	Logical negation



# Comparison operators (relational operators)

Operator	Description
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

# Compound assignment operators

Operator	Example	Equal to
=	a = 10	a = 10
+=	a += 10	a = a + 10
-=	a -= 10	a = a - 10
*=	a *= 10	a = a * 10
/=	a /= 10	a = a / 10
// =	a //= 10	a = a // 10
%=	a %= 10	a = a % 10
**=	a **= 10	a = a ** 10
&=	a &= 10	a = a & 10
=	a  = 10	a = a   10
^=	a ^= 10	a = a ^ 10
>>=	a >>= 10	a = a >> 10
<<=	a <<= 10	a = a << 10

# Identity operators

Operator	Description
<b>is</b>	It return true if two variable point the same object and false otherwise.
<b>is not</b>	It return false if two variable point the same object and true otherwise.

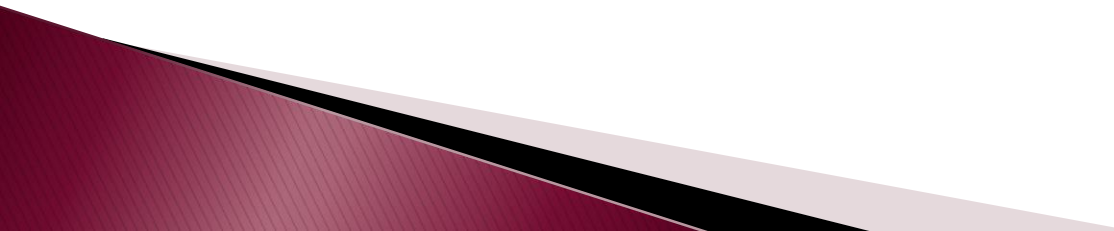
# Membership Operators

Operator	Description
<b>in</b>	It returns true if value/variable is found in the sequence and false otherwise.
<b>not in</b>	It returns true if value/variable is not found in the sequence and false otherwise.

# Truth Table

A	B	$\sim A$	$A \& B$	$A \mid B$	$A \wedge B$
		not A	A and B	A or B	[A xor B]
0	0	1	0	0	0
0	1	1	0	1	1
1	0	0	0	1	1
1	1	0	1	1	0

# Basic algorithm structures

- ▶ Sequence
    - set of instructions
  - ▶ Selection
    - if-else statements
  - ▶ Iteration
    - loops
- 

# Sequence

- ▶ the simplest control mechanism is actually a set of instructions executed in a predetermined sequence
- ▶ the instructions are written on a separate line

instructions<sub>1</sub>

instructions<sub>2</sub>

...

instructions<sub>n</sub>

# Conditional statement (branching, selection)

- ▶ examine the condition and execute different instruction depending on the truth value statement

if condition:

statement1

[else:

statement2]

- ▶ The expression in the condition must always return a logical value, or the result must be convertible to a boolean.
- ▶ If the condition is true, it executes statement1, otherwise if there is an else branch, it executes statement2, if there is no else branch, it does nothing if the condition is not true.



# Conditional statement

**if** condition:

statement1

statement2

**else:**

statement3

statement4

# Exercise

- ▶ Write a program which determines if the input number is even or odd.

# Solution

```
x = int(input("x="))
if x % 2 == 0:
    print("Even!")
else:
    print("Odd!")
```

# Exercise

- ▶ Input an integer number and decide whether it ends in 4 or not.

# Solution

```
n = int(input("n = "))  
if n % 10 == 4:  
    print("ends with 4.")  
else:  
    print("not ends with 4")
```

# Exercise

- ▶ Write a program which determines if a triangle can be constructed from three segments.  
If it is possible, give the area of the triangle.

# Solution

```
import math

a = int(input("a = "))
b = int(input("b = "))
c = int(input("c = "))
if a + b > c and a + c > b and b + c > a:
    print("The triangle can be constructed!")
    p=(a+b+c)/2
    A =math.sqrt(p * (p - a) * (p - b) * (p - c))
    print("A = {0:.2f}".format(A))
else:
    print("The triangle cannot be constructed!")
```

# Conditional statement

```
if condition1:  
    statement1  
elif condition2:  
    statement2  
elif condition3:  
    satement3  
...  
...  
else:  
    statementn
```

- ▶ the else branch can be omitted
- ▶ any number of elif branches can be used



# Conditional statement

- ▶ the program will execute the first branch whose condition is satisfied, or if no condition is satisfied and there is an else branch, it will execute the instruction(s) there.
- ▶ if several logical expressions are concatenated in a condition using AND / OR operators:
  - **AND (and)** operator, if the first expression is false, then the second is ignored, the result will be false anyway.
  - **OR (or)** operator, if the first expression is true, the second expression is not worth dealing with, the result will be true anyway.

# Exercise

- ▶ Write a program that reads two integers, the program prints out if they are equal or if not, which one is greater.

# Solution

```
a = int(input("a="))
b = int(input("b="))
if a == b:
    print("Equal!")
elif a > b:
    print("Greater: ", a)
else:
    print("Greater: ", b)
```

# Exercise

- ▶ Ask the user "How many hours?".
- ▶ The program will greet the user according to the number read:
  - Good Morning (5:00–11:59),
  - Good Afternoon (12:00–17:59),
  - Good Evening (18:00–21:59),
  - Good Night (22:00–4:59)

# Solution

```
hour = int(input('hour = '))

if hour > 5 and hour < 12:
    print("Good Morning!")
elif hour >= 12 and hour < 18:
    print("Good Afternoon!")
elif hour >= 18 and hour < 22:
    print("Good Evening!")
elif (hour >= 22 and hour <= 24) or (hour >= 0 and
hour <= 5):
    print("Good Night!")
else:
    print("Wrong value!")
```

# The Pass statement

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

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

# 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:**
  - $5 + 2 = 7$
  - $5 - 2 = 3$
  - $5 * 2 = 10$
  - $5 / 2 = 2.5$
  - $5 // 2 = 2$
  - $5 \% 2 = 1$

# 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:**  $x_1, x_2$

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

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



# 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, 9 -> 31
    - 2023, 4 -> 30
    - 2020, 2 -> 28
    - 2001, 2 -> 29