Chapter 2: Elementary Programming

Instructor: Dr. Murat Tunc

Lecture 2

November 16th, 2021

Last Week (Summary)

"Hello World!" Program

```
# This program prints Hello World!

print("Hello World!")
```

Statement

- A **statement** represents an action or a sequence of actions
- The statement **print**(**“Hello World!”**) in the program is a statement to display the greeting “Hello World!”

```
# This program prints Hello World!

print("Hello World!")
```

Comments

- Line 1 (in **green color**) is a **comment** that documents what the program is and how it is constructed

- They are not programming statements, and thus are **ignored** by the compiler

```
# This program prints Hello World!

print("Hello World!")
```

Special Symbols

- ( ) i.e. Opening and closing parentheses
  - Used with functions and methods
- # i.e. Pound sign
  - Precedes a comment line
- “ ” i.e. Opening and closing double quotation marks
  - Enclosing a string (i.e. a series of characters)

```
# This program prints Hello World!

print("Hello World!")
```

Programming Errors

- 1) Syntax Errors
  - **Detected** by the compiler
- 2) Logic Errors
  - Produce **incorrect results**

Programming Errors

Syntax Error

```
# This program prints Hello World!

print("Hello World!)
```

"

Programming Errors

Logic Error

```
# This program prints the average of 3 + 4

print("Average of 3 and 4 is ")

print(3 + 4 / 2)
```

Output: Average of 3 and 4 is 5
Correct output: 3.5

Correct way: (3+4)/2 = 3.5

Chapter 2: Elementary Programming

Instructor: Dr. Murat Tunc

Lecture 2

November 16th, 2021

In-class Exercise 1 (Group study – 10 min)

Write a program that

1) **reads in an input** as the **radius** of a circle from the user, and
2) **calculates** and **prints** the **area of a circle**

Writing a Simple Program

- **Designing Algorithm**: how a problem is solved by listing the actions that need to be taken
  - Description can be in natural language or in pseudocode

- Algorithm to calculate area of a circle:
  - **Step 1:** Read in the circle's radius from the user
  - **Step 2:** Compute area using the formula:

$$\text{area} = \pi * \text{radius} * \text{radius}$$

  - **Step 3:** Display the result

Writing a Simple Program

- Translating the algorithm into a program

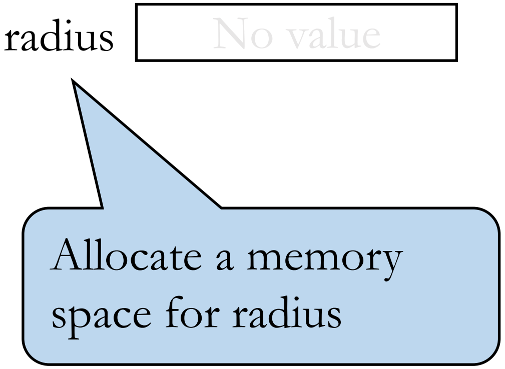
```
# Step 1: Read in radius from the user

# Step 2: Compute area

# Step 3: Display the area
```

Writing a Simple Program

```
# Step 1: Read in radius from the user
radius = input("Please input the radius of a circle and press Enter: ")
radius = float(radius)

# Step 2: Compute area
area = radius * radius * 3.14159

# Step 3: Display the area
print("The area of a circle with the radius", radius, "is", area)
```

Tracing a Program Execution

```
    # Step 1: Read in radius from the user
    radius = input("Please input the radius of a circle 
and press Enter: ")
    radius = float(radius)

    # Step 2: Compute area
    area = radius * radius * 3.14159

    # Step 3: Display the area
    print("The area of a circle with the radius", radius, 
"is", area)
```

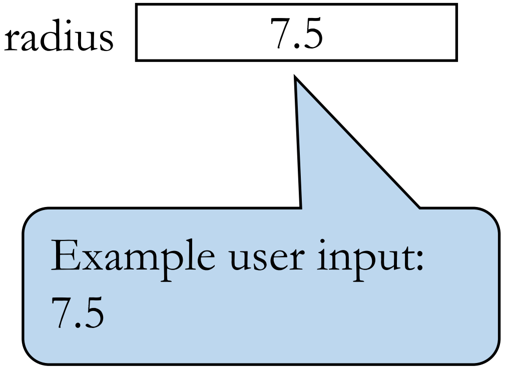


radius | No value

Allocate a memory space for radius

Tracing a Program Execution

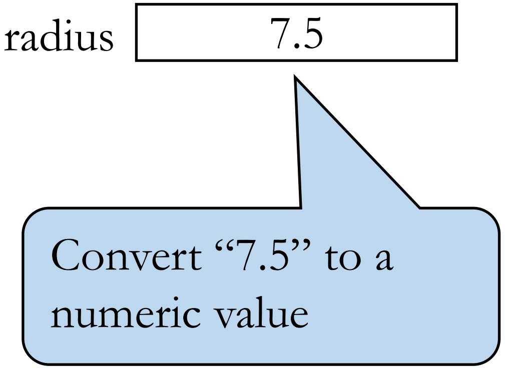
```
# Step 1: Read in radius from the user
radius = input("Please input the radius of a circle and press Enter: ")
radius = float(radius)

# Step 2: Compute area
area = radius * radius * 3.14159

# Step 3: Display the area
print("The area of a circle with the radius", radius, "is", area)
```



Example user input:
7.5

Tracing a Program Execution

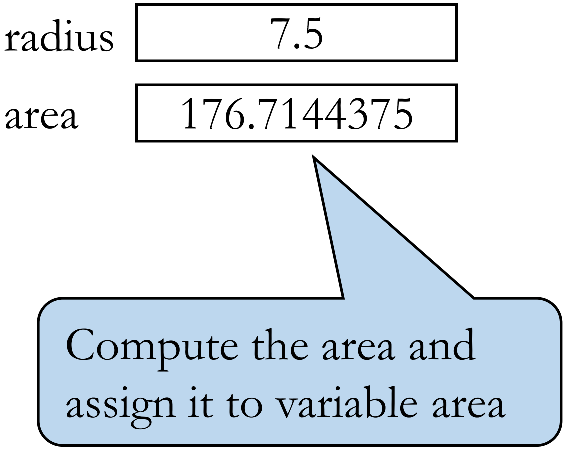
```
    # Step 1: Read in radius from the user
    radius = input("Please input the radius of a circle
and press Enter: ")
    radius = float(radius)

    # Step 2: Compute area
    area = radius * radius * 3.14159

    # Step 3: Display the area
    print("The area of a circle with the radius", radius,
"is", area)
```



Convert "7.5" to a numeric value

Tracing a Program Execution

```
    # Step 1: Read in radius from the user
    radius = input("Please input the radius of a circle
and press Enter: ")
    radius = float(radius)

    # Step 2: Compute area
    area = radius * radius * 3.14159

    # Step 3: Display the area
    print("The area of a circle with the radius", radius,
"is", area)
```



radius	7.5
area	176.7144375

Compute the area and assign it to variable area

Tracing a Program Execution

```
# Step 1: Read in radius from the user
radius = input("Please input the radius of a circle and press Enter: ")
radius = float(radius)

# Step 2: Compute area
area = radius * radius * 3.14159

# Step 3: Display the area
print("The area of a circle with the radius", radius, "is", area)
```

radius	7.5
area	176.7144375

Display the area

In-class Exercise 2 (Self study - 10 minutes)

Write a program that

1) **reads** in a **Celsius** degree from the user,
2) **converts** Celsius to **Fahrenheit** degree, and
3) **displays** the result

Hint. Fahrenheit = (9 / 5) * Celsius + 32

In-class Exercise 2 - Answer

```
# Step 1: Read in Celsius degree from the user
celsius = input("Please input the Celsius degree and press Enter: ")
celsius = float(celsius)

# Step 2: Convert Celsius to Fahrenheit degree
fahrenheit = (9 / 5) * celsius + 32

# Step 3: Display the result
print("Celsius degree of", celsius, "is equal to", fahrenheit, "Fahrenheit degree")
```

Review

- Q: **input**( ) statement reads in a value from the user as numeric.

  A. True

  B. False

- Ans: B

- Q: What does the following program prints?

```
radius = 7.5
print(“radius”)
```

A. 7.5
B. radius

- Ans: B

- Q: What does the following program prints?

  ```
  radius = 7.5
  print(radius)
  ```

  A. 7.5
  B. radius

- Ans: A

In-class Exercise 3 (Practice at home – 10 min)

Write a program that

1) **reads three numbers** from the user and
2) **displays their average**

Identifiers

- **Identifiers** are the names that identify the elements such as variables, constants, methods, classes, and packages in a program

- An identifier is a sequence of characters that **consist of letters**, **digits**, and **underscores** (_).

- An identifier must start with a letter, an underscore (_). It **cannot start with a digit**

Identifiers

- An identifier **cannot** **be a reserved keyword**
  - **import, return, except, if, else, …**

- An identifier **cannot** be **True** or **False**

- An identifier can be of any length

- Python is case sensitive
  - Area, area, and AREA are all **different** identifiers

Variables

- The program needs to read the radius entered by the user from the keyboard. This raises two important issues:
  - Reading the radius
  - **Storing** the radius in the program

- In order to store the radius, the program needs to declare a symbol called a **variable**

Variables

- Variables are used to **store values** to be used later in a program

- They are called variables because their **values can be changed**

- We need to tell the compiler the name of the variable

- Choose descriptive names for variables
  - **radius** for radius
  - **area** for area

Assignment Statements

- We can assign a value to a variable by using an **assignment statement**
- In Python, the **equal sign** (=) is used as the assignment operator
- The syntax for assignment statements is as follows:

variable = expression

- An expression represents a computation involving values, variables, and operators that taking them together, evaluates to a value

Assignment Statements

- x = 1 # Assign 1 to x

- radius = 7.5 # Assign 7.5 to radius

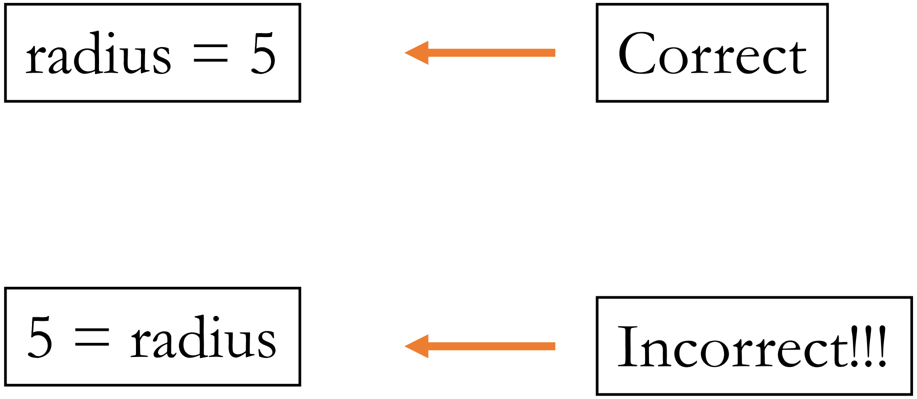
- a = “A” # Assign “A” to a

- count = 2 # Assign value 2 to count

  count = count + 1 # Assign addition of count
  # and 1 to count

Assignment Statements

- To assign a value to a variable, you must place **the variable name to the left** of the assignment operator



radius = 5

Correct

5 = radius

Incorrect!!!

Review

- Q: Which of the following are valid identifiers?

A. a

B. +app

C. 3number

D. radiusOfTheCircle

E. $2

F. d+7

G. True

- Ans: A, D

Numeric Literals

- A literal is a constant value that appears directly in a program

- For **example**, 34 and 0.305 are literals in the following statements

```
numberOfYears = 34
weight = 0.305
```

Numeric Literals

- An **integer** literal can be assigned to a variable
  - integerVariable = 3
  - **print** (**type** (**integerVariable**)) **# Displays <class 'int'>**

- A **floating point** literal written with a decimal point
  - floatVariable = 3.14
  - **print** (**type** (**floatVariable**)) **# Displays <class 'float'>**

Numeric Literals - Conversion

- We can convert a floating point literal to an integer literal
  - Removes the decimal parts of a float number
- **Example:**

```
numberBeforeConvertion = 3.14
numberAfterConvertion = int( numberBeforeConvertion )
print ( type ( numberAfterConvertion ) )
    # Displays <class 'int'>
print ( numberAfterConvertion )
    # Displays 3
```

Numeric Literals - Conversion

- Similarly, we can convert an integer literal to a float number
  - Simply adds a decimal point and a zero
- **Example:**

```
numberBeforeConvertion = 3
numberAfterConvertion = float( numberBeforeConvertion )
print ( type ( numberAfterConvertion ) )
    # Displays <class 'float'>
print ( numberAfterConvertion )
    # Displays 3.0
```

Numeric Operations

Name	Meaning	Example	Result
+	Addition	34 + 1	35
-	Subtraction	34.0 – 0.1	33.9
*	Multiplication	300 * 30	9000
/	Division	1.0 / 2.0	0.5
%	Remainder	20 % 3	2

Division, Integer Division and Remainder

- **Division** operator: **/**
  - will **always** result in a floating point number
  - **Example:** 5 / 2 yields a floating point number 2.5
- **Integer division** operator: **//**
  - **Example:** 5 // 2 yields an integer number 2
- **Remainder** operator: **%**
  - will result in the **remainder** of the division
  - **Example:** 5 % 2 yields an integer number 1

- Remainder operation is useful in programming
  - **Even** number % 2 is always 0
  - **Odd** number % 2 is always 1

Division, Integer Division and Remainder

- The **result of a division** operation is **always** a floating point number
    - 4 / 2   **# Results in 2.0**

- The result of an **integer division** and **remainder** operation
    - **Depends** on the types of the numeric literals used in the operations

Division, Integer Division and Remainder

- If **at least one floating point number** is used in integer division and remainder operations
  - The **result** will be a **floating point number**
- **Examples:**
  - 7 // 3.0 **# Results in 2.0**
  - 7.0 % 3 **# Results in 1.0**

Division, Integer Division and Remainder

- If **two integer numbers** are used in integer division and remainder operations
  - The **result** will be **an integer number**
- **Examples:**
  - 7 // 3 **# Results in 2**
  - 7 % 3 **# Results in 1**

In-class Exercise 4 (Self-study – 10 min)

Write a program to **obtain minutes** and **remaining seconds** from an amount of **time in seconds**.

1) **Read in the time in seconds** from the user (**Example**: 200 seconds)

2) **Convert** 200 seconds => 3 minutes and 20 seconds

In-class Exercise 4 - Answer

```
    # Step 1: Read in the time in seconds from the user
    timeInSeconds = float( input("Please input the time (in
seconds) and press Enter: ") )

    # Step 2: Convert the time to minutes and seconds
    minutes = int ( timeInSeconds // 60 )
    seconds = timeInSeconds % 60

    # Step 3: Display the result
    print(timeInSeconds, "seconds equals to", minutes,
“minutes and", seconds, “seconds")
```

Review

- count = 7 / 3 **# What is the value stored in count?**
  A. 1
  B. 2
  C. 2.3333

- **Ans:** 2.3333

- test = 7 % 3 **# What is the value stored in test?**
  A. 1
  B. 2
  C. 2.3333

- **Ans:** 1

- count = 7 // 3 **# What is the value stored in count?**

  A. 1

  B. 2

  C. 2.3333

- **Ans:** 2

- test = 7.5 // 3 **# What is the value stored in test?**

  A. 2.5

  B. 2

  C. 2.0

- **Ans:** 2.0

Exponent Operations

- **pow** (a, b) is used to compute a^b

```
print(pow(2, 3))
# Displays 8
print(pow(4, 0.5));
# Displays 2.0
print(pow(2.5, 2));
# Displays 6.25
print(pow(2.5, -2));
# Displays 0.16
```

Arithmetic Expressions

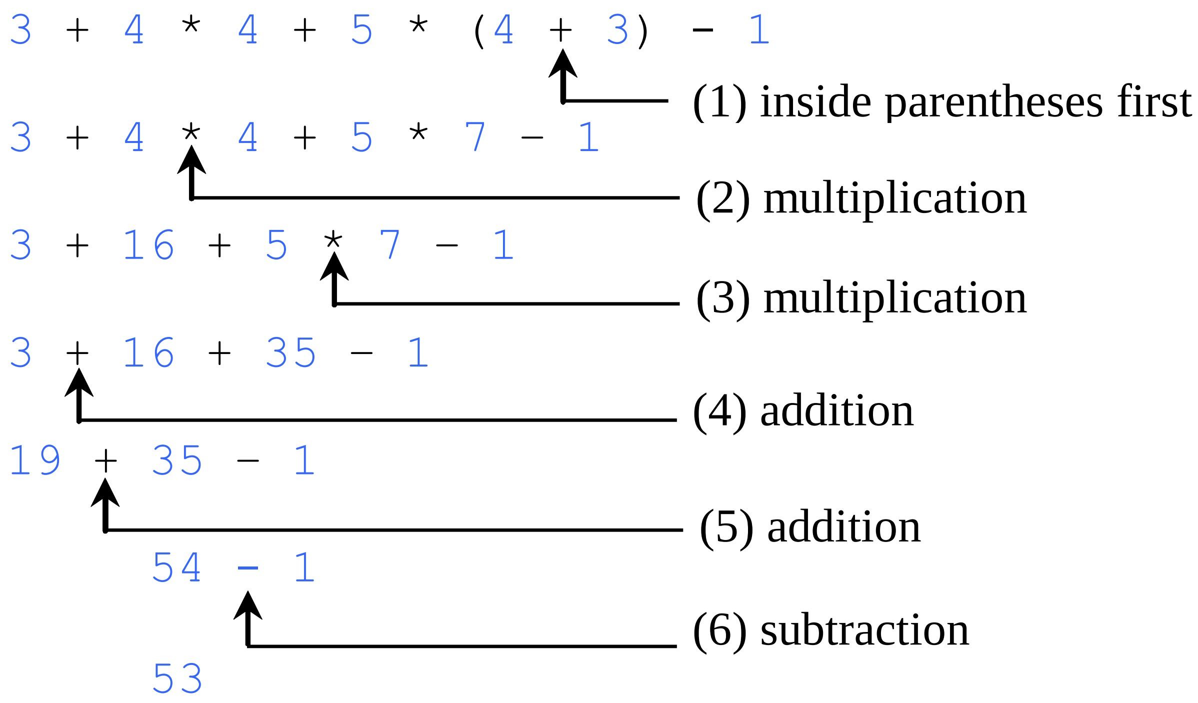
$$\boxed{\frac{3+4x}{5}-\frac{10(y-5)(a+b+c)}{x}+9(\frac{4}{x}+\frac{9+x}{y})}$$

is translated to

```
(3+4*x)/5 – 10*(y-5)*(a+b+c)/x + 9*(4/x + (9+x)/y)
```

How to Evaluate an Expression

- We can safely apply the arithmetic rule for evaluating a Python expression



```
3 + 4 * 4 + 5 * (4 + 3) - 1
```

(1) inside parentheses first

```
3 + 4 * 4 + 5 * 7 - 1
```

(2) multiplication

```
3 + 16 + 5 * 7 - 1
```

(3) multiplication

```
3 + 16 + 35 - 1
```

(4) addition

```
19 + 35 - 1
```

(5) addition

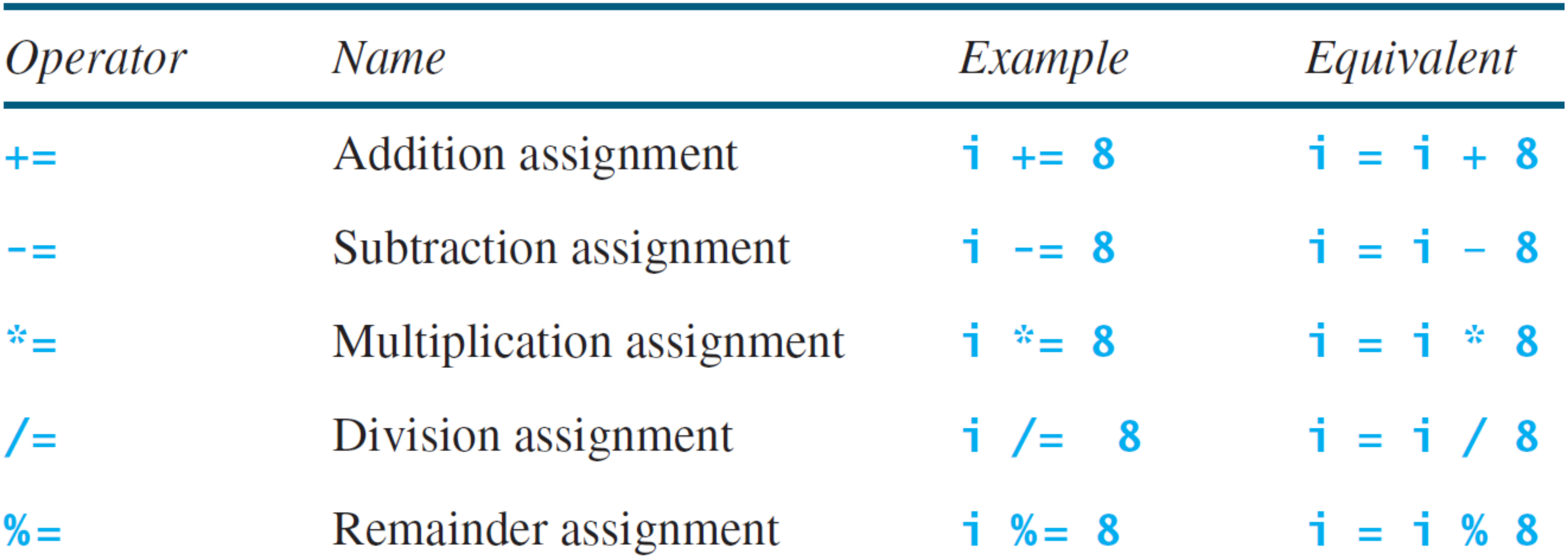
```
54 - 1
```

(6) subtraction

```
53
```

Augmented Assignment Operators

- The operators +, -, *, /, and % can be combined with the assignment operator (=) to form **augmented operators**



Operator	*Name*	*Example*	*Equivalent*
`+=`	Addition assignment	`i += 8`	`i = i + 8`
`-=`	Subtraction assignment	`i -= 8`	`i = i - 8`
`*=`	Multiplication assignment	`i *= 8`	`i = i * 8`
`/=`	Division assignment	`i /=  8`	`i = i / 8`
`%=`	Remainder assignment	`i %= 8`	`i = i % 8`

Review

-
  ```
  x = 5
  x /= 2          # What is the value stored in x?
  ```

  A. 2
  B. 3
  C. 2.5
  D. Error

- **Ans:** C

```
test = 5
test += test + 1          # What is the value in test?
```

A. 6

B. 5

C. 11

D. Error

- **Ans:** C

```
x = 5
x //= 2            # What is the value stored in x?
```

A. 2
B. 3
C. 2.5
D. Error

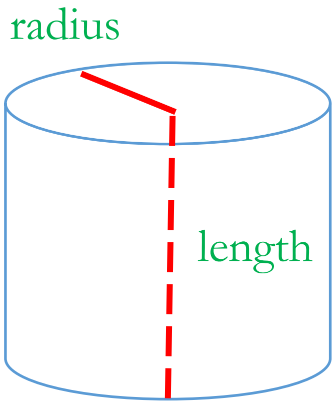
- **Ans:** A

Practice Question 1

Write a program that

1) **reads a two digit integer** from the user and
2) **swap its digits** to create a new integer.

For example, if an integer is 93, after swapping it becomes 39.



Practice Question 2

Write a program that

1) **reads numbers for radius and length** from the user and

2) **displays the volume of a cylinder** on console.

area = radius * radius * π

volume = area * length

Practice Question 3

Write a program that

1) **reads the values of x and y** from the user and
2) **display the following result** on console.

$$y^{x-7} + \frac{x+y}{4} - \frac{2\,(x-y)+3}{5} + \frac{y}{3\,x\ -10}$$

Check the result for x=10, y=5 (The answer should be 126.4)