Unit 3. Presentation Slides

Learning Python

Display a message

- The **print()** function is used to display messages on the screen.

```python
helloworld.py

print("Hello, world!")
```

Variables and Data Types

- Variables **store information** that can be used later.

```python
variables_types.py

name = "Alice"   # String
age = 13         # Integer
height = 1.6     # Float
is_student = True  # Boolean
```

User input

- The input() function allows users to **enter information**.

```
                    input.py

name = input("What is your name? ")
print("Hello, " + name + "!")
```

Basic operations

- Python can perform **mathematical operations** like addition, subtraction, multiplication, and division.

```
                    math.py
x = 10
y = 3

print(x + y)   # Addition
print(x - y)   # Subtraction
print(x * y)   # Multiplication
print(x / y)   # Division
print(x % y)   # Modulus (remainder)
print(x ** y)  # Exponentiation
```

Conditional Statements

- The if statement allows you to **make decisions** in your code.

```
●●●                    decisions.py

age = 13
if age >= 18:
    print("You are an adult.")
else:
    print("You are a minor.")
```

Loops: *for* loop

- Loops help **repeat** a block of code multiple times.

```
●●●                    for_loop.py

for i in range(5):
    print("Iteration", i)
```

Loops: *while* loop

- Loops help **repeat** a block of code multiple times.

```
● ● ●                  while_loop.py

count = 0
while count < 5:
    print("Count is", count)
    count = count + 1
```

Functions

- In Python, **functions** are like little helpers that perform specific tasks.
- They make **code organized** and reusable.
- Let's explore different **types** of functions.

```
● ● ●                  f_sayhello.py

# Function definition
def say_hello():
    print("Hello, world!")

# Function Call
say_hello()
```

Functions

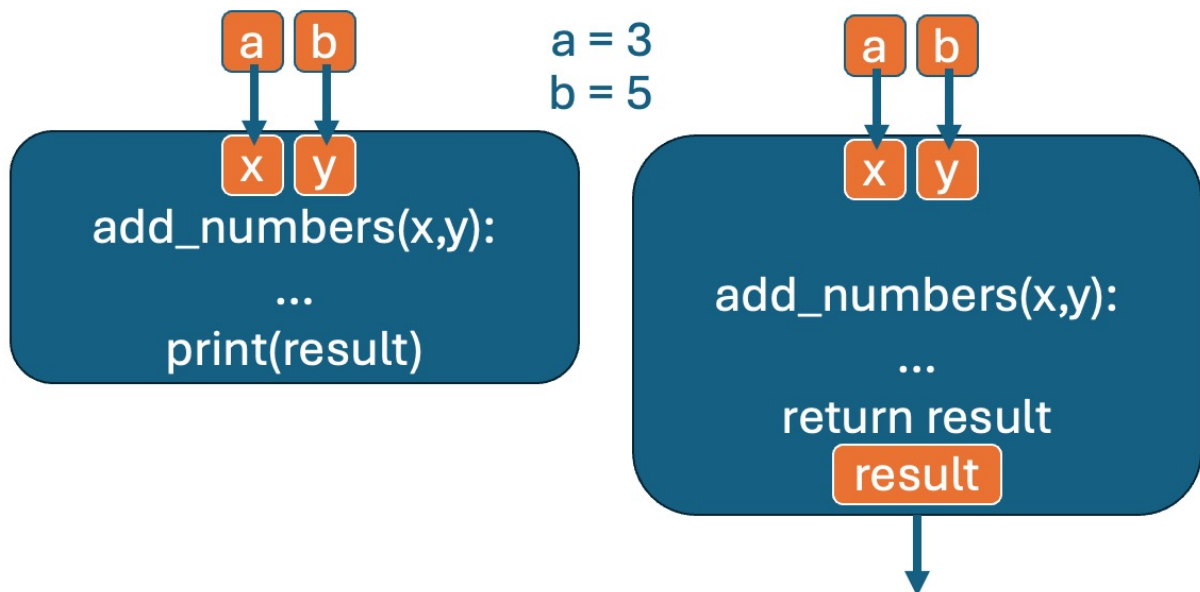- Functions allow us to **reuse code** by defining reusable blocks.

```python
def greet(name):
    print("Hello, " + name + "!")

greet("Alice")
greet("Bob")
```

Functions with/without return value

Function: *add_numbers* (without return value)

- This Python code **defines** and **calls** a function that adds two numbers and prints the result.
- This is a function <u>without</u> a **return value**.

```python
# add_numbers.py

# Function definition
def add_numbers(x, y):
    result = x + y
    print("Sum:", result)

# Function call
add_numbers(5, 3)
```

Function: *add_numbers* (with return value)

- This Python code **defines** and **calls** a function that adds two numbers and prints the result.
- This is a function <u>**with**</u> a **return value**.

```python
# add_numbers_r.py

# Function definition
def add_numbers(x, y):
    result = x + y
    return result

# Function call and return value
sum_xy = add_numbers(5, 3)
# Print result
print("Sum:", sum_xy)
```

Function: *get_average*

- This Python code defines a function that **calculates the average** of two numbers and **returns** the result.

```python
                                    get_average.py

# Function definition
def getAverage(x1, x2):
    x = (x1 + x2) / 2
    return x
# Function call
average = getAverage(6,4)
print(average)
```

Function: getTextGrade

- This Python code defines a function that **converts** a numerical grade into a text-based evaluation (e.g., "Very good", "Good", or "Fail").

```python
                                    get_text_grade.py

# Function definition
def getTextGrade(ngrade):
    text = ""
    if ngrade >= 7.5:
        text = "Very good"
    elif 5 <= ngrade < 7.5:
        text = "Good"
    else:
        text = "Fail"
    return text

#Function call
t = getTextGrade(8)
print("Final grade: ", t)
```

Programming, Artificial Intelligence and Robotics II

Presentation

2024/2025 · 3ESO

Enrique Benimeli