

# Wiki - Week one | An Introduction to Interactive Programming in Python

## Functions — Functions

- Functions are reusable pieces of programs that take an input and produce an output.
- A function definition is a compound statement consisting of a header and a body.
- The header includes the keyword `def`, a sequence of parameters enclosed by parentheses, followed by a colon `:`.
- The body consists of a sequence of statements, all indented by 4 spaces.
- Functions may return a value using the keyword `return` or have a side effect (e.g., `print`).
- To evaluate a function call, replace the function's parameters in the body of the function by their associated values in the call and execute the body of the function.
- Lecture examples - [Functions](#)
- More examples - [Structure of Functions](#), [Uses of Functions](#), [Scope of Variables](#), [Examples of Functions](#)

## Indentation — Functions

- Indentation consists of whitespace formed by blanks, tabs, and newlines.
- Leading white space indicates indentation level (4 spaces per level) and specifies logical grouping of statements in Python.
- Incorrect indentation can lead to errors.
- Lecture examples - [Functions](#)
- More examples - [Function Errors](#)

## Remainders and modular arithmetic — More Operations

- Standard long division yields a quotient and a remainder. The integer division operator `//` computes the quotient. The operator `%` computes the remainder.
- For any integers `a` and `b`,  $a == b * (a // b) + (a \% b)$ .
- In Python, `a \% b` always returns an answer that is between 0 and `b` (even if `a` and/or `b` is negative).
- Remainders and modular arithmetic are very useful in games for the purpose of "wrapping" the canvas, i.e; causing objects that pass off of one side of the canvas to reappear on the opposite side of the canvas.
- Lecture examples - [More operations](#)
- More examples - [Modulus](#), [Math Module](#),

## Modules — More Operations

### Boolean Expressions — Logic and Comparisons

- The constants `True` and `False` of the type `bool`.
- These constants can be combined to form Boolean expressions via the logical operators `and`, `or`, and `not`.

- The and of two Boolean expressions is True if both of the expressions are True.
- The or of two Boolean expressions is True if at least one of the expressions is True.
- Lecture examples - None
- More examples - [Booleans](#), [Boolean Logic](#)

## **Relational Operators** — Logic and Comparisons

- The values of two arithmetic expressions can be compared using the operators ==, !=, <, >, <=, >=.
- These comparisons return either True or False .
- Lecture examples - None
- More examples - [Comparison](#), [Boolean Expressions](#)

## **Conditional Statements** — Conditionals

- Conditional statements are compound statements consisting one or more clauses headed by the keywords if, elif, and else.
- Each if or elif clause is followed by a Boolean expression and a colon :.
- If the Boolean expression for a clause is True, the body of the clause is executed.
- Lecture examples - [Conditionals](#)
- More examples - [if-elif-else](#), [Examples of Conditionals](#)

## **Programming Tips** — [Week 1](#)