# Study Guide for Computer Science 1

This study guide covers fundamental computer science concepts using Python. It is designed to help you prepare for your exam, which consists of multiple-choice questions and programming problems.

---

## 1. Introduction to Programming

### \*\*Programming – General\*\*

The process of designing and building an executable computer program to accomplish a specific computing result.

### \*\*Program\*\*

A set of instructions written in a programming language that a computer can execute to perform a specific task.

### \*\*Input, Output, Process\*\*

- \*\*Input\*\*: Data received by the program from the user or another system.

- \*\*Process\*\*: Operations performed on the input data to produce a result.

- \*\*Output\*\*: The result produced by the program and provided to the user or another system.

### \*\*Variables\*\*

Named storage locations in memory that hold data which can be modified during program execution. Variables have a name, type, and value.

### \*\*Python Interpreter\*\*

A program that reads and executes Python code, translating it into machine code line by line.

### \*\*Statement\*\*

A single line of code that performs an action.

```python

print("Hello, World!") # This is a print statement

```

### \*\*Expression\*\*

A combination of values, variables, and operators that can be evaluated to produce another value.

```python

result = 2 + 3 # The expression '2 + 3' evaluates to 5

```

### \*\*Comments\*\*

Non-executable lines in the code used to explain or annotate the code. In Python, comments start with `#`.

```python

# This is a comment

```

---

## 2. Data Types and Structures

### \*\*Python Data Types\*\*

- \*\*String\*\*: A sequence of characters enclosed within single (`'`) or double (`"`) quotes.

```python

text = "Hello, World!"

```

- \*\*Integer\*\*: Whole numbers, positive or negative, without decimals.

```python

count = 42

```

- \*\*Float\*\*: Numbers that contain a decimal point.

```python

pi = 3.14

```

### \*\*Container Types\*\*

- \*\*List\*\*: An ordered, mutable collection of items.

```python

my\_list = [1, 2, 3]

```

- \*\*Dictionary\*\*: A collection of key-value pairs.

```python

my\_dict = {'name': 'Alice', 'age': 30}

```

- \*\*Set\*\*: An unordered collection of unique items.

```python

my\_set = {1, 2, 3}

```

- \*\*Tuple\*\*: An ordered, immutable collection of items.

```python

my\_tuple = (1, 2, 3)

```

### \*\*Mutable / Immutable\*\*

- \*\*Mutable\*\*: Objects that can be changed after creation (e.g., lists, dictionaries, sets).

- \*\*Immutable\*\*: Objects that cannot be changed after creation (e.g., strings, integers, floats, tuples).

### \*\*Key-Value Pairs\*\*

Elements in a dictionary where each key is associated with a value, allowing for fast retrieval by key.

---

## 3. Operators and Expressions

### \*\*Arithmetic Operators\*\*

- \*\*Addition (`+`)\*\*

- \*\*Subtraction (`-`)\*\*

- \*\*Multiplication (`\*`)\*\*

- \*\*Division (`/`)\*\*: Divides and returns a float.

- \*\*Floor Division (`//`)\*\*: Divides and returns an integer by discarding the fractional part.

- \*\*Modulo (`%`)\*\*: Returns the remainder of a division.

- \*\*Exponentiation (`\*\*`)\*\*: Raises a number to the power of another.

### \*\*Comparison Operators\*\*

- \*\*Equal to (`==`)\*\*

- \*\*Not equal to (`!=`)\*\*

- \*\*Greater than (`>`)\*\*

- \*\*Less than (`<`)\*\*

- \*\*Greater than or equal to (`>=`)\*\*

- \*\*Less than or equal to (`<=`)\*\*

### \*\*Logical Operators\*\*

- \*\*And (`and`)\*\*: Returns `True` if both operands are true.

- \*\*Or (`or`)\*\*: Returns `True` if at least one operand is true.

- \*\*Not (`not`)\*\*: Returns `True` if the operand is false.

### \*\*Relational Operators\*\*

Another term for comparison operators.

### \*\*Order of Operations\*\*

The sequence in which operations are performed in an expression, following the rules of precedence (e.g., parentheses, exponents, multiplication/division, addition/subtraction).

### \*\*Increment/Decrement\*\*

Increasing or decreasing the value of a variable.

```python

x += 1 # Increment x by 1

x -= 1 # Decrement x by 1

```

### \*\*Escape Sequences\*\*

- `\n`: Newline

- `\t`: Tab

- `\\`: Backslash

- `\'`: Single quote

- `\"`: Double quote

- \*\*Usage Example\*\*:

```python

print("Line1\nLine2") # Outputs two lines

```

---

## 4. Control Flow

### \*\*Conditionals\*\*

- \*\*If Statement\*\*: Executes a block of code if a condition is true.

```python

if condition:

# code to execute

```

- \*\*Elif Statement\*\*: Checks another condition if the previous conditions were false.

```python

elif another\_condition:

# code to execute

```

- \*\*Else Statement\*\*: Executes a block of code if all previous conditions were false.

```python

else:

# code to execute

```

### \*\*Types of Loops\*\*

- \*\*For Loop\*\*: Iterates over a sequence (such as a list, tuple, or string).

```python

for item in sequence:

# code to execute

```

- \*\*While Loop\*\*: Repeats a block of code as long as a condition is true.

```python

while condition:

# code to execute

```

### \*\*Loop Examples\*\*

- \*\*For Loop Example\*\*:

```python

for i in range(5):

print(i)

```

- \*\*While Loop Example\*\*:

```python

count = 0

while count < 5:

print(count)

count += 1

```

---

## 5. Other Concepts

### \*\*Identifier\*\*

A name given to entities like variables, functions, classes, etc., to identify them in the code. Must start with a letter or underscore, followed by letters, digits, or underscores.

```python

my\_variable = 10

```

### \*\*Keywords\*\*

Reserved words in Python that have special meanings and cannot be used as identifiers.

- Examples: `if`, `else`, `for`, `while`, `def`, `class`, `import`, etc.

### \*\*Whitespace\*\*

Spaces, tabs, and newlines in code. In Python, indentation (whitespace at the beginning of a line) is significant and defines code blocks.

### \*\*Errors\*\*

- \*\*Syntax Error\*\*: Occurs when the code violates Python's grammar rules, making it impossible to parse.

```python

print("Hello World" # Missing closing parenthesis

```

- \*\*Runtime Error\*\*: Occurs during program execution, often due to invalid operations.

```python

result = 10 / 0 # Division by zero error

```

- \*\*Logic Error\*\*: The program runs without crashing but produces incorrect results due to flaws in the algorithm.

---

## 6. Container Types – Details

### \*\*Lists\*\*

- \*\*Creation\*\*:

```python

my\_list = [1, 2, 3]

```

- \*\*Access Elements\*\*:

```python

first\_item = my\_list[0]

```

- \*\*Common Methods\*\*:

- `append(item)`: Adds an item to the end.

```python

my\_list.append(4)

```

- `insert(index, item)`: Inserts an item at a specified index.

```python

my\_list.insert(1, 'a')

```

- `remove(item)`: Removes the first occurrence of an item.

```python

my\_list.remove(2)

```

- `pop(index)`: Removes and returns the item at the specified index.

```python

item = my\_list.pop(0)

```

- `len(my\_list)`: Returns the number of items.

```python

length = len(my\_list)

```

### \*\*Dictionaries\*\*

- \*\*Creation\*\*:

```python

my\_dict = {'key1': 'value1', 'key2': 'value2'}

```

- \*\*Access Values\*\*:

```python

value = my\_dict['key1']

```

- \*\*Common Methods\*\*:

- `keys()`: Returns a list of keys.

```python

keys = my\_dict.keys()

```

- `values()`: Returns a list of values.

```python

values = my\_dict.values()

```

- `items()`: Returns a list of key-value pairs.

```python

items = my\_dict.items()

```

- `get(key)`: Returns the value for the specified key.

```python

value = my\_dict.get('key1')

```

### \*\*Sets\*\*

- \*\*Creation\*\*:

```python

my\_set = {1, 2, 3}

```

- \*\*Common Methods\*\*:

- `add(item)`: Adds an item.

```python

my\_set.add(4)

```

- `remove(item)`: Removes an item.

```python

my\_set.remove(2)

```

- \*\*Set Operations\*\*:

- \*\*Union\*\* (`|`): Combines items from both sets.

```python

set1 | set2

```

- \*\*Intersection\*\* (`&`): Items common to both sets.

```python

set1 & set2

```

- \*\*Difference\*\* (`-`): Items in one set but not the other.

```python

set1 - set2

```

### \*\*Tuples\*\*

- \*\*Creation\*\*:

```python

my\_tuple = (1, 2, 3)

```

- \*\*Access Elements\*\*:

```python

first\_item = my\_tuple[0]

```

- \*\*Note\*\*: Tuples are immutable; elements cannot be added or removed after creation.

---

\*\*Remember\*\*: Practice writing Python code using these concepts to prepare for the programming portion of your exam.