Module 1i: Introduction to Problem Solving and Python Fundamentals

Premanand S

Assistant Professor,
School of Electronics and Engineering,
Vellore Institute of Technology, Chennai

premanand.s@vit.ac.in

July 19, 2025

Keywords

What Are Keywords in Python?

- Keywords are **reserved words** in Python with predefined meanings.
- They form the **syntax and structure** of the Python language.
- Cannot be used as **variable names, function names, or identifiers**.

Why are Keywords Important?

- They form the grammar rules of the language.
- Used to create conditions, loops, function definitions, exception handling, etc.
- They cannot be used as variable names because they're already defined for specific purposes.

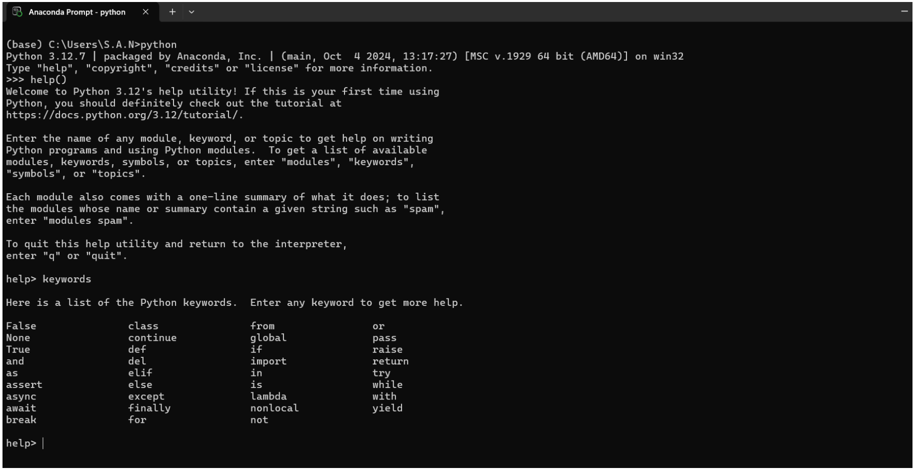
Keywords - Importance

- **Value Keywords:** True, False, None
- **Operator Keywords:** and, or, not, in, is
- **Control Flow Keywords:** if, elif, else
- **Iteration Keywords:** for, while, break, continue, else
- **Structure Keywords:** def, class, with, as, pass, lambda
- **Returning Keywords:** return, yield
- **Import Keywords:** import, from, as
- **Exception-Handling Keywords:** try, except, raise, finally, else, assert
- **Asynchronous Programming Keywords**: async, await
- **Variable Handling Keywords:** del, global, nonlocal
- Understanding their proper use is key to improving your skills and knowledge of Python.

How do keywords work?

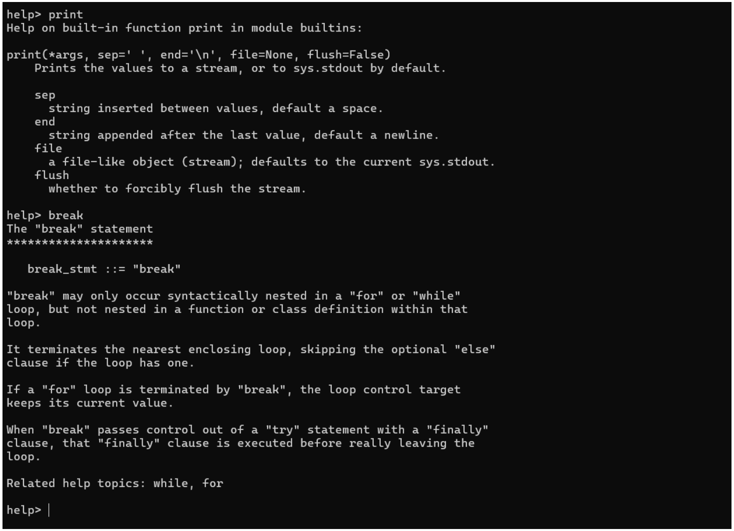
- Python automatically understands and executes keywords according to its internal grammar rules.
- They are case-sensitive (e.g., True is valid, but true is not).

How to Get All Python Keywords Programmatically?



```
(base) C:\Users\S.A.N>python
Python 3.12.7 | packaged by Anaconda, Inc. | (main, Oct  4 2024, 13:17:27) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> help()
Welcome to Python 3.12's help utility! If this is your first time using
Python, you should definitely check out the tutorial at
https://docs.python.org/3.12/tutorial/.

Enter the name of any module, keyword, or topic to get help on writing
Python programs and using Python modules.  To get a list of available
modules, keywords, symbols, or topics, enter "modules", "keywords",
"symbols", or "topics".

Each module also comes with a one-line summary of what it does; to list
the modules whose name or summary contain a given string such as "spam",
enter "modules spam".

To quit this help utility and return to the interpreter,
enter "q" or "quit".

help> keywords

Here is a list of the Python keywords.  Enter any keyword to get more help.

False               class               from                or
None                continue            global              pass
True                def                 if                  raise
and                 del                 import              return
as                  elif                in                  try
assert              else                is                  while
async               except              lambda              with
await               finally             nonlocal            yield
break               for                 not

help> |
```

How to Get All Python Keywords Programmatically?



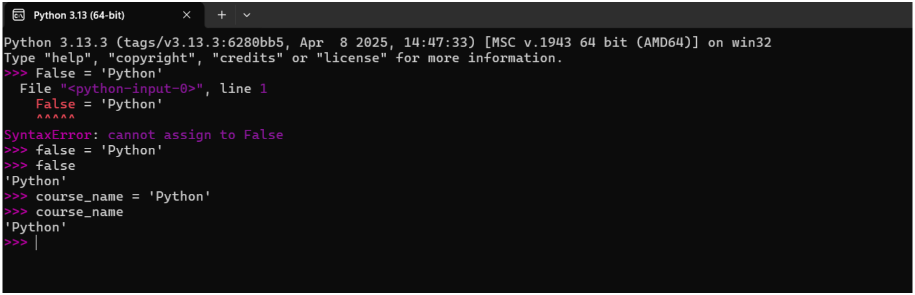
```
help> print
Help on built-in function print in module builtins:

print(*args, sep=' ', end='\n', file=None, flush=False)
    Prints the values to a stream, or to sys.stdout by default.

    sep
      string inserted between values, default a space.
    end
      string appended after the last value, default a newline.
    file
      a file-like object (stream); defaults to the current sys.stdout.
    flush
      whether to forcibly flush the stream.

help> break
The "break" statement
*********************

   break_stmt ::= "break"

"break" may only occur syntactically nested in a "for" or "while"
loop, but not nested in a function or class definition within that
loop.

It terminates the nearest enclosing loop, skipping the optional "else"
clause if the loop has one.

If a "for" loop is terminated by "break", the loop control target
keeps its current value.

When "break" passes control out of a "try" statement with a "finally"
clause, that "finally" clause is executed before really leaving the
loop.

Related help topics: while, for

help> |
```

How to Get All Python Keywords Programmatically?

Example (Python Snippet)

```python
import keyword

print(keyword.kwlist)
print("Total keywords:", len(keyword.kwlist))
```

Keyword not as Identifiers



```
Python 3.13.3 (tags/v3.13.3:6280bb5, Apr  8 2025, 14:47:33) [MSC v.1943 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> False = 'Python'
  File "<python-input-0>", line 1
    False = 'Python'
    ^^^^^
SyntaxError: cannot assign to False
>>> false = 'Python'
>>> false
'Python'
>>> course_name = 'Python'
>>> course_name
'Python'
>>> 
```

Summary: Keywords in Python

- **What:** Predefined reserved words that define Python's grammar.
- **Why:** Enable logical structure, control flow, and code modularity.
- **When/Where:** Used across all parts of Python scripts and programs.
- **How:** Use `import keyword` to access and explore keywords.

Tip: Keep keywords in mind when naming variables to avoid syntax errors.

Identifiers

What are Identifiers in Python?

- Identifiers are the **names used to identify variables, functions, classes, modules, and objects**.
- Examples: `x`, `studentName`, `total_marks`, `calculateArea`

Why and Where Are Identifiers Used?

- **Why:**
  - To reference data and functions in code.
  - To give meaningful names that improve readability and maintenance.
- **Where:**
  - Anywhere you declare variables, define functions, classes, loops, parameters, etc.

Rules for Naming Identifiers

- Can contain letters (A–Z, a–z), digits (0–9), and underscores (_)
- **Must not start with a digit**
- Cannot be a **Python keyword**
- Are **case-sensitive** (e.g., `total` and `Total` are different)
- Avoid using special characters (e.g., , $, %, etc.)
- Use lowercase letters for variable names and functions, and separate words with underscores (snake_case). For example, calculate_average, what_is_your_name
- Use uppercase letters for constants. For example, MAX_SCORE
- Use CamelCase (or PascalCase) for class names. For example, StudentRecord.

Examples of Valid and Invalid Identifiers

Valid Identifiers

- `name`, `student_1`, `_temp`, `calcTotal`

Invalid Identifiers

- `1name` – starts with a digit
- `total$` – contains special character
- `for` – reserved keyword

Assignments – Identifiers

- Create 3 valid identifiers for: name, age, and city. Print them.
- Try using an invalid identifier and observe the error.
- Create variables with similar names (e.g., `mark`, `Mark`) and test case-sensitivity.
- List 5 keywords and try to use them as variable names. What error do you get?
- Write a program that checks if a given string is a valid identifier
- Create a function that takes 5 variable names from user and prints which ones are valid identifiers.

print() function

What is print()?

- The print() function is a built-in Python function that outputs data to the console (standard output).

Why is it used?

- To display information to the user.
- For debugging, by printing variable values or messages.
- To format output for better readability.

When to use it?

- When showing results after a computation.
- For printing messages or data during program flow.
- To inspect variable values during development or testing.

Syntax - print()

Example (Python Snippet)

```
print(*objects, sep=' ', end='\n', file=sys.stdout,
flush=False)
```

How does it work?

Example (Python Snippet)

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

name = "python"
age = 34
print(name, age)
```

Print - Modifiers

Example (Python Snippet)

```
print('She said:"Hello" and then left')

print("She said: 'Hello' and then left")

print("She said:\"Hello\" and then left")
```

Multiple line to print

Example (Python Snippet)

```
print('Hello world!')
print('Hello world!')
print('Hello world!')
?
```

Multiple line to print

Example (Python Snippet)

```
print('Hello world!\n Hello world! \n Hello world!')
```

Concatenation

Example (Python Snippet)

```
print('Hello' + 'Python')
print('Hello'+' '+'Python')
print('Hello' + ' Python')
print('Hello'+ '      Python')
```

Spacing

Example (Python Snippet)

```
print('Hi guys! did you notice something?')
  print('?')    #one space or indentation space ?


     print('Did you see now?') # new cell
```

print() - paramaters

Example (Python Snippet)

```
print("Python", "3.12", sep="-->", end="!!!\n",
flush=True)

with open("log.txt", "w") as f:
    print("Logging to file!", file=f)

import time
for i in range(5):
    print(i, end=' ', flush=True)
    time.sleep(1)
```

print() - paramaters

Example (Python Snippet)

```
# Using sep and end parameters

print("2025", "06", "29", sep="-")

print("Hello", end=" ")
print("World")
```

Escape Characters and the print() Function in Python

- The `print()` function displays output to the console.
- Escape characters start with a backslash '\' and are used to insert special characters in strings.
- Common escape sequences:
  - `\n` – Newline
  - `\t` – Tab
  - `\'` – Single Quote
  - `\"` – Double Quote
  - `\\` – Backslash

Examples

```
print("Hello\nWorld")          # Newline
print("Name:\tAlice")          # Tab
print('It\'s sunny')           # Escape single quote
print("She said \"Hi\"")       # Escape double quote
print("C:\\Users\\Admin")      # Escape backslash
```

In writing the file?

Example (Python Snippet)

```
#  Creating and Printing with file output

with open("output.txt", "w") as file:
    print("Hello File!", file=file)
```

Debug it!

Example (Python Snippet)

```
print(Day1 - String Manipulation’)
print(’String concatenation is done with ’+’ sign’)
print(’eg. print(’hello’+’world’)’)
print((’newlines can be created with a backslash and n’)
```

Solution - Debug it!

Example (Python Snippet)

```
print('Day1 - String Manipulation')
print("String concatenation is done with '+' sign")
print('eg. print(\'hello'+'world\')')
print('newlines can be created with a backslash and n')
```

Assignments

1. Print your name, age, department, and favorite language (one per line).
2. Print: `Python is easy!` five times in one line, separated by ` | `.
3. Print the triangle pattern:
   - `*`
   - `**`
   - `***`
4. Use variables to print: `5 + 3 = 8`

Assignments

- Accept user's name, age, and city. Display a formatted message: *Example: Hello Python, you are 34 years old and live in Chennai.*
- Print today's date in the format `YYYY-MM-DD` using the `sep='-'` parameter.
- Use string multiplication to print a row of 5 equal signs: =====
- Print the sentence with embedded double quotes: `"Python" is a powerful language.`
- Join the words `code`, `debug`, `run` with arrow symbols using `sep` and `end`:
  Expected output: `code --> debug --> run.`

Assignments

- Use a loop with `flush=True` to print a live countdown from 5 to 1.
- Redirect the output of `print()` to a file named `output.txt`.
- Simulate a loading bar: `Loading [ ]` 50% using time delays.
- Display a column-aligned subject-marks table using `\t` or formatting:

```
Subject      Marks
Math         89
Physics      92
```

input() function

What is `input()`?

- `input()` is a built-in function in Python.
- It is used to take **user input** during program execution.
- It always returns a **string** by default.

Why and Where to Use `input()`

- **Why?**
  - Makes programs interactive.
  - Allows dynamic data entry instead of hardcoding.
- **Where/When?**
  - In forms, calculators, quizzes, menus.
  - When user decisions or inputs are required at runtime.

Syntax and Notes

Syntax:

```
variable = input("Prompt message")
```

Note:

- Always returns data as `str`.
- Use `int()`, `float()` to convert input to numeric types if needed.

Examples

Example 1 – Basic String Input:

```
name = input("Enter your name: ")
print("Hello,", name)
```

Example 2 – Numeric Input with Typecasting:

```
num = int(input("Enter a number: "))
print("Square is:", num * num)
```

Assignments – `input()` Function

- Take your name as input and greet the user.
- Accept a number and print its cube.
- Take name, age, city and format them into a sentence.
- Input 2 numbers, display their sum, product, and average.
- Ask a user to input a sentence and print it in reverse.
- Take 5 numbers (one-by-one) and compute the average.
- Create a contact form (name, email, phone) and print it as a table.

Input vs Print functions

Example (Python Snippet)

```
print('What is your name?')

input('What is your name?')
```

Input + print + comment functions

Example (Python Snippet)

```
#printing by getting the data from user
print('Hello' +' '+ input('What is your name?'))
```

Variables

Variables

- Variables are used to store data that can be referenced and manipulated in a program.
- A variable is essentially a name given to a data value, and once you assign a value to a variable, you can use that variable name to access the stored value.
- Dynamically typed
- Assigning values
- Case-sensitive
- Naming conventions as like Identifiers

Importance of variables

Example (Python Snippet)

```
input('Whats your contact number?')

python_sir = input('Whats your contact number?')

print(python_sir)
```

Why the name - VARIABLE?

Example (Python Snippet)

```
name = 'Prem'
name = 'Anand'
name = 'Premanand'
print(name)
```

Why the name - VARIABLE?

Example (Python Snippet)

```
name = 'Prem'
name = 38
name = 'Python'
name = 2.0
print(name)
```

Len function

Len function

- The len() function in Python is used to determine the length (i.e., the number of items) of an object.
- The object can be a sequence (such as a string, list, tuple, or range) or a collection (such as a dictionary, set, or frozen set).

Len functions

Example (Python Snippet)

```
string = "Hello, world!"
print(len(string))
```

Print + input + Len functions

Example (Python Snippet)

```
name = input('Whats your name?')
print(len(name))

or

print(len(input('whats your name?')))
```

Comments

Comments

- Important while writing program
- Describes about the content of the segment of program or whole program
- # - symbol for comments
- Python interpreter ignores the comment line
- Types : Single line & Multi-line

Single line comment

Example (Python Snippet)

```
# This is a single-line comment
x = 10  # This is an inline comment
```

Multi-line comment

Example (Python Snippet)

```
# This is a multi-line comment
# that spans multiple lines
# using consecutive single-line comments.
name = 'Python'
professional = 'Engineering'
```

Multi-line comment

Example (Python Snippet)

```
"""
This is a multi-line comment
that spans multiple lines
using triple quotes.
"""
```

Docstring

Docstring

- Docstrings are a special kind of comment used to document modules, functions, classes, and methods.
- hey are written using triple quotes and can span multiple lines.
- Docstrings are accessible at runtime using the __doc__ attribute.

Docstring

Example (Python Snippet)

```
def add(a, b):
    """
    This function adds two numbers and returns the result.

    Parameters:
    a (int): The first number.
    b (int): The second number.

    Returns:
    int: The sum of the two numbers.
    """
    return a + b

print(add.__doc__)
```

Feature	**Comments**	**Docstrings**
Purpose	Explain specific lines or blocks of code	Document modules, classes, methods, and functions
Syntax	# for single-line comments	Triple quotes (''') or (""") for multi-line
Multi-line Format	Consecutive # or triple quotes	Triple quotes
Runtime Access	Not accessible at runtime	Accessible via `__doc__` attribute
Placement	Anywhere in the code	First statement within a module, class, method, or function
Length	Typically brief	Can be detailed and extensive

Table: Comparison of Comments and Docstrings in Python

Statement

Satement

- Statement is a unit of code that performs an action.
- Each statement in Python is executed by the interpreter, and they form the building blocks of Python programs

Statement - Types

- Expression Statements: x=5
- Assignment Statements: num1 = 25
- Control Flow Statements: if statements: if condition
- Control Flow Statements: for loops: for item in iterable
- Control Flow Statements: while loops: while condition
- Function statement: def my_function(): pass
- Class statement: class my_class(): pass
- Import statement: import math or from math import sqrt
- Return statement: return num1
- Break statement: break
- Continue statement: continue
- Pass statement: pass

Expression Statement

Example (Python Snippet)

```python
print("Hello, World!")
```

Assignment Statement

Example (Python Snippet)

```
number = 10
```

Control Flow Statement (if statement):

Example (Python Snippet)

```
if number > 5:
    print("Number is greater than 5")
```

Loop Statement (for loop):

Example (Python Snippet)

```
for i in range(5):
    print(i)
```

Function Definition:

Example (Python Snippet)

```
def greet(name):
    print(f"Hello, {name}!")
```

Class Definition:

Example (Python Snippet)

```
class Person:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print(f"Hello, my name is {self.name}")
```

Import Statement:

Example (Python Snippet)

```
import math
print(math.sqrt(16))
```

Return Statement:

Example (Python Snippet)

```
def add(a, b):
    return a + b
```

Break and Continue Statements:

Example (Python Snippet)

```
for i in range(10):
    if i == 5:
        break  # Exit the loop when i equals 5
    print(i)

for i in range(10):
    if i % 2 == 0:
        continue  # Skip even numbers
    print(i)
```

Pass Statement:

Example (Python Snippet)

```
def empty_function():
    pass  # Placeholder for future code
```

Indentation

Indentation

- Indentation in Python is a critical part of the syntax and is used to define the structure and flow of the code.
- Unlike many other programming languages that use braces or keywords to define blocks of code, Python uses indentation levels to determine the grouping of statements.

Indentation - Basic Example

Example (Python Snippet)

```
if True:
    print("This is indented to show it is part of if block.")
    print("This line is also part of the if block.")
print("This line is not indented, so it’s outside if block.")
```

Indentation - Function Definition

Example (Python Snippet)

```python
def greet(name):
    print(f"Hello, {name}!")
    print("Welcome to the Python tutorial.")
```

Indentation - Class Definition

Example (Python Snippet)

```
class Person:
    def __init__(self, name):
        self.name = name

    def say_hello(self):
        print(f"Hello, my name is {self.name}")
```

Indentation - Loops

Example (Python Snippet)

```
for i in range(5):
    print(i)
    if i % 2 == 0:
        print(f"{i} is even.")
```

Indentation - Conditional Statements

Example (Python Snippet)

```
x = 10
if x > 5:
    print("x is greater than 5.")
    if x < 15:
        print("x is also less than 15.")
```

Problem1: Swapping the strings from users data, by using temporary variable

Example (Python Snippet)

```
Input:
string1 = 'Hello'
string2 = 'World'

Output:
string1 = 'World'
string2 = 'Hello'
```

Answer1: Swapping the strings from users data, by using temporary variable

Example (Python Snippet)

```
string1 = input('Whats the first string?')
string2 = input('Whats the second string?')

string3 = string1
string1 = string2
string2 = string3

print('First string:', string1)
print('Second string:', string2)
```

QUIZ 1

Example (Understanding)

```
1. Which line of Python code is valid?
a. var a = 2
b. a = 12
c. a:12
d. 12 = a
```

QUIZ 2

Example (Understanding)

```
2. Which is the best variable name for players1 username?
a. p1 user name = 'ghilli_boyz'
b. 1_player_username = 'ghilli_boyz'
c. player1_username = 'ghilli_boyz'
d. ply1 = 'ghilli_boyz'
```

Problem2: Write a Python program

Example (Apply print, input, variables)

You're working on a fun project to generate unique band names. The program asks the user for the name of the city they grew up in, their pet's name, and their favorite music genre. It then combines these inputs to suggest a potential band name.

Answer2: Band name generation!

Example (Apply print, input, variables)

```
print('Welcome to the band name generator!')
city = input('What is the name of the city you grew up in?\n'
pet_name = input('What is the name of your pet?\n')
music_genre = input('What is your favorite music genre?\n')
print('Your band name could be ' + city + '_' + pet_name +
'_' + music_genre)
```

mail me: er.anandprem@gmail.com / premanand.s@vit.ac.in
ring me: +91 73586 79961
Follow me: Linkedin
Medium Blogs
Analytics Vidhya: Blogs

Don't just code — think, plan, and solve