IDENTIFIERS OF PYTHON

- An identifier is used to identify a variable, function, class, or other object in a program. It is a sequence of one or more letters, digits, or underscores, with the first character being a letter or underscore.
- Identifiers are case-sensitive, meaning that '**my_var**' and '**My_Var**' are two different identifiers.
- Identifiers cannot use reserved words or keywords, such as **if**, **while**, **for**, and **print**, as they have special meaning in the Python language.
- Identifiers can be any length, but it is a good practice to keep them concise and descriptive to make the code more readable and understandable.

DIFF BTW VARIABLES & IDENTIFIERS

```
# This is an example of an identifier
pi = 3.14
```

In this example, pi is an identifier that represents the value of pi (3.14). It is a variable identifier, because it is used to store a value in memory.

```
# This is another example of an identifier
def multiply(a, b):
    return a * b
```

In this example, 'multiply' is an identifier that represents a function that takes two arguments and returns their product. It is a function identifier, because it is used to define a function in Python.

In both cases, pi and multiply are examples of identifiers that are used to refer to different types of objects in Python. The first one is a variable identifier, while the second one is a function identifier.

DIFFERENT CASE TYPES

- camelCase - In this camel case, the first word of the identifier is lowercase and the first letter of each subsequent word is capitalized. There are no underscores between words.

  Example: studioWebsite, flowWebsite

- PascalCase: In PascalCase, the first letter of each word in the identifier is capitalized, and there are no underscores between words. This convention is also known as "UpperCamelCase" .

  Example: StudioWebsite, FlowWebsite

- snake_case : In snake_case, all words in the identifier are lowercase and separated by underscores. This convention is typically used for naming variables and functions in Python.

  Example: studio_website, flow_website

Rules of Identifiers

- Identifiers may start with an alphabet.
- Identifiers can start with single leading underscore (ex: _name = "karthik").
- Identifiers can not start with a number.
- Keywords can not be used as identifiers.
- Built-In functions can not be used as identifiers.

Types of Identifiers

Private Identifier - (ex: _salary = 47000, _acctNo = 6658133357)

An identifier with a single leading underscore (e.g., **_variable**) is conventionally used to indicate that it is intended to be a private variable or method. This means that it is intended for use only within the current module and not by code outside of the module.

However, it's important to note that this is just a convention, and it does not actually prevent code outside of the module from accessing or modifying the variable.

Strong Private Identifier/Magical Methods - (ex: __priv_func(), __init__(self))

double underscores (also known as "dunder" or "magic" methods) added as a prefix to a variable name is used to define a class attribute or method as private. This means that the attribute or method can only be accessed within the class itself, and not from outside the class.

The purpose of using double underscores as a prefix is to protect class attributes and methods from accidental or deliberate modification from outside the class.

LITERALS

In programming, a literal is a notation used to represent a fixed value in source code. In other words, it's a way to directly specify a value within the code itself, without having to calculate or compute it at runtime.

For example, in Python, the number 42 is a numeric literal, the string "Hello, world!" is a string literal, and the Boolean value True is a Boolean literal.

We can use type() built in function to find the exact type of literals.

Example:

name = "ram"

print(type(name)) will be <class 'str'> which means string, likewise we can find exact type of all literals

OPERATORS

Arithmetic operators [+, -, *, /, %, //]

- % this operator is named as Modulus, it gives the remainder of division.
- // this operator is named as Floor division, it gives the rounded value of division.

Logical operators [AND, OR, NOT]

Assignment operators [=,+=,-=,*=,/=,%=,//=]

Relational operators [==, =, <=, >=, <, >]

Membership operators [in, not in]

Identity operators [is, is not]

COMMENTS AND QUOTATIONS

Comments are used to describe what the line exactly does inside a program.

Types:

- Single line comments
- Multi line comments

Quotations are used to represent string values in python

Types:

- Single quotation
- Double quotations
- Triple single quotation
- Triple double quotations

[NOTE: anything inside triple quotations in python is called as doc strings which represents extra documentation purpose of a program]

Keywords in python are used to invoke reserved actions.
List of 33 keywords used in python are,

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