Python

1. Variables/Identifiers
2. Keywords
3. Operators
4. Python Data Types

1.1. Variables/Identifiers

Rules

- A Python variable name must start with a letter or the underscore character.
- A Python variable name cannot start with a number.
- A Python variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _).
- Variable in Python names are case-sensitive (name, Name, and NAME are three different variables).
- The reserved words(keywords) in Python cannot be used to name the variable in Python.

In [1]:
```
1 _alpha=6
2 print(_alpha)
```

```
6
```

In [2]:
```
1 1alpha=6
2 print(1alpha)
```

```
  Cell In[2], line 1
    1alpha=6
    ^
SyntaxError: invalid decimal literal
```

In [3]:
```
1 ap12_jfd=7
2 print(ap12_jfd)
```

```
7
```

In [4]:
```
1 apha@l =3
```

```
  Cell In[4], line 1
    apha@l =3
    ^
SyntaxError: cannot assign to expression here. Maybe you meant '==' instea
d of '='?
```

In [5]:
```
a = 5
print(a)
```

```
5
```

In [6]:
```
a==5
```

Out[6]:
```
True
```

In [7]:
```
Num=6
```

In [8]:
```
num=5
```

In [9]:
```
print(Num)
print(num)
```

```
6
5
```

In [10]:
```
a=5
b=6
```

In [11]:
```
a+b
```

Out[11]:
```
11
```

1.2. Keywords

- Python Keywords are some predefined and reserved words in Python that have special meanings.
- Keywords are used to define the syntax of the coding.
- The keyword cannot be used as an identifier, function, or variable name.
- All the keywords in Python are written in lowercase except True and False.

Keywords	Description
and	This is a logical operator which returns true if both the operands are true else returns false.
or	This is also a logical operator which returns true if anyone operand is true else returns false.
not	This is again a logical operator it returns True if the operand is false else returns false.
if	This is used to make a conditional statement.
elif	Elif is a condition statement used with an if statement. The elif statement is executed if the previous conditions were not true.
else	Else is used with if and elif conditional statements. The else block is executed if the given condition is not true.
for	This is used to create a loop.
while	This keyword is used to create a while loop.
break	This is used to terminate the loop.
as	This is used to create an alternative.
def	It helps us to define functions.

Keywords	Description
lambda	It is used to define the anonymous function.
pass	This is a null statement which means it will do nothing.
return	It will return a value and exit the function.
True	This is a boolean value.
False	This is also a boolean value.
try	It makes a try-except statement.
with	The with keyword is used to simplify exception handling.
assert	This function is used for debugging purposes. Usually used to check the correctness of code
class	It helps us to define a class.
continue	It continues to the next iteration of a loop
del	It deletes a reference to an object.
except	Used with exceptions, what to do when an exception occurs
finally	Finally is used with exceptions, a block of code that will be executed no matter if there is an exception or not.
from	It is used to import specific parts of any module.
global	This declares a global variable.
import	This is used to import a module.
in	It's used to check whether a value is present in a list, range, tuple, etc.
is	This is used to check if the two variables are equal or not.
none	This is a special constant used to denote a null value or avoid. It's important to remember, 0, any empty container(e.g empty list) do not compute to None
nonlocal	It's declared a non-local variable.
raise	This raises an exception.
yield	It ends a function and returns a generator.
async	It is used to create asynchronous coroutine

In [ ]:
```
1
```

1.3. Operators

Operators in general are used to perform operations on values and variables. These are standard symbols used for the purpose of logical and arithmetic operations. In this article, we will look into different types of Python operators.

- **OPERATORS**: These are the special symbols. Eg- + , * , /, etc.
- **OPERAND**: It is the value on which the operator is applied.

Types of Operators in Python

1. Arithmetic Operators
2. Comparison Operators
3. Logical Operators
4. Bitwise Operators
5. Assignment Operators

6. Identity Operators and Membership Operators

1.3.1. Arithmetic Operators

Python Arithmetic operators are used to perform basic mathematical operations like addition, subtraction, multiplication, and division.

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power: Returns first raised to power second	x ** y

Precedence of Arithmetic Operators in Python

The precedence of Arithmetic Operators in python is as follows:

P – Parentheses
E – Exponentiation
M – Multiplication (Multiplication and division have the same precedence)
D – Division
A – Addition (Addition and subtraction have the same precedence)
S – Subtraction

In [12]:
```
1 2**2
```

Out[12]: 4

In [13]:
```
1 2+3
```

Out[13]: 5

In [14]:
```
1 5-3
```

Out[14]: 2

In [15]:
```
1 5*9
```

Out[15]: 45

In [16]:
```
1 10/3
```

Out[16]: 3.3333333333333335

In [17]:
```
1 10//3
```

Out[17]: 3

In [18]:
```
-10//3
```

Out[18]: -4

In [19]:
```
5%2
```

Out[19]: 1

In [20]:
```
25&3
```

Out[20]: 1

In [21]:
```
a = 10
if a % 2 == 0:
    print('even no.')
else:
    print('odd no.')

```

```
even no.
```

In [ ]:
```

```

In [ ]:
```

```

1.3.2. Comparison Operators in Python

In Python Comparison of Relational operators compares the values. It either returns True or False according to the condition.

Operator	Description	Syntax
>	Greater than: True if the left operand is greater than the right	x > y
<	Less than: True if the left operand is less than the right	x < y
==	Equal to: True if both operands are equal	x == y
!=	Not equal to – True if operands are not equal	x != y
>=	Greater than or equal to True if the left operand is greater than or equal to the right	x >= y
<=	Less than or equal to True if the left operand is less than or equal to the right	x <= y

```
= is an assignment operator and == comparison operator.
```

In [22]:
```
5>4
```

Out[22]: True

In [23]:
```
5<3
```

Out[23]: False

In [24]:
```
a=3
```

In [25]:
```
1 a==4
```

Out[25]:
```
False
```

In [26]:
```
1 a!=3
```

Out[26]:
```
False
```

In [ ]:
```
1
```

1.3.3. Logical Operators in Python

Python Logical operators perform Logical AND, Logical OR, and Logical NOT operations. It is used to combine conditional statements.

Operator	Description	Syntax
and	Logical AND: True if both the operands are true	x and y
or	Logical OR: True if either of the operands is true	x or y
not	Logical NOT: True if the operand is false	not x

In [27]:
```
1 not (5>4 or 6>8)
```

Out[27]:
```
False
```

1.3.4. Bitwise Operators in Python

Python Bitwise operators act on bits and perform bit-by-bit operations. These are used to operate on binary numbers.

Operator	Description	Syntax
&	Bitwise AND	x & y
|	Bitwise OR	x | y
~	Bitwise NOT	~x
^	Bitwise XOR	x ^ y
>>	Bitwise right shift	x>>
<<	Bitwise left shift	x<<

In [28]:
```
1 5>4 | 6>8
```

Out[28]:
```
False
```

In [ ]:
```
1
```

1.3.5. Assignment Operators in Python

Python Assignment operators are used to assign values to the variables.

Operator	Description	Syntax
=	Assign the value of the right side of the expression to the left side operand	x = y + z
+=	Add AND: Add right-side operand with left-side operand and then assign to left operand	a+=b , a=a+b

Operator	Description	Syntax
-=	Subtract AND: Subtract right operand from left operand and then assign to left operand	a-=b , a=a-b
*=	Multiply AND: Multiply right operand with left operand and then assign to left operand	a*=b , a=a*b
/=	Divide AND: Divide left operand with right operand and then assign to left operand	a/=b , a=a/b
%=	Modulus AND: Takes modulus using left and right operands and assign the result to left operand	a%=b , a=a%b
//=	Divide(floor) AND: Divide left operand with right operand and then assign the value(floor) to left operand	a//=b , a=a//b
**=	Exponent AND: Calculate exponent(raise power) value using operands and assign value to left operand	a**=b , a=a**b
&=	Performs Bitwise AND on operands and assign value to left operand	a&=b , a=a&b
^=	Performs Bitwise xOR on operands and assign value to left operand	a^=b , a=a^b
>>=	Performs Bitwise right shift on operands and assign value to left operand	a>>=b , a=a>>b

In [29]:
```
a=a+b
a+=b
```

1.3.6. Identity Operators in Python

In Python, is and is not are the identity operators both are used to check if two values are located on the same part of the memory. Two variables that are equal do not imply that they are identical.

```
is          True if the operands are identical
is not      True if the operands are not identical
```

1.3.6. Membership Operators in Python

In Python, in and not in are the membership operators that are used to test whether a value or variable is in a sequence.

```
in            True if value is found in the sequence
not in        True if value is not found in the sequence
```

In [30]:
```
# Identity Operators
a = 10
b = 20
c = a

print(a is not b)
print(a is c)
```

```
True
True
```

In [31]:
```python
# Membership Operators
x = 24
y = 20
list = [10, 20, 30, 40, 50]

if (x not in list):
    print("x is NOT present in given list")
else:
    print("x is present in given list");

if (y in list):
    print("y is present in given list")
else:
    print("y is NOT present in given list")
```

```
x is NOT present in given list
y is present in given list
```

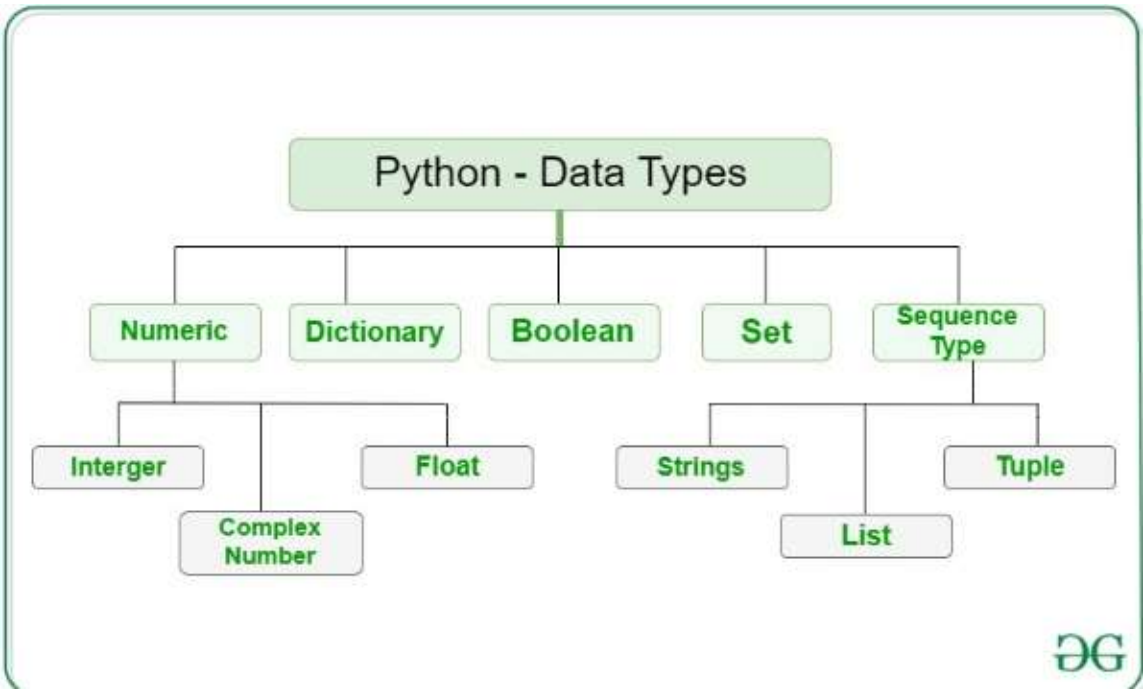
In [ ]:
```python

```

1.4. Python Data Types

Data types are the classification or categorization of data items. It represents the kind of value that tells what operations can be performed on a particular data. Since everything is an object in Python programming.

- Numeric
- Sequence Type
- Boolean
- Set
- Dictionary
- Binary Types



1.4.1. Numeric

- **Integers –** This value is represented by int class. It contains positive or negative whole numbers (without fractions or decimals). In Python, there is no limit to how long an integer value can be.
- **Float –** This value is represented by the float class. It is a real number with a floating-point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.
- **Complex Numbers –** Complex number is represented by a complex class. It is specified as (real part) + (imaginary part)j. For example – 2+3j

In [32]:
```
a = 5
print("Type of a: ", type(a))

b = 5.0
print("\nType of b: ", type(b))

c = 2 + 4j
print("\nType of c: ", type(c))
```

```
Type of a:  <class 'int'>

Type of b:  <class 'float'>

Type of c:  <class 'complex'>
```

1.4.2. Sequence Data Type in Python

The sequence Data Type in Python is the ordered collection of similar or different data types. Sequences allow storing of multiple values in an organized and efficient fashion. There are several sequence types in Python –

- Python String
- Python List
- Python Tuple

String

In [33]:
```
ram is a good "boy".
```

```
  Cell In[33], line 1
    ram is a good "boy".
              ^
SyntaxError: invalid syntax
```

In [34]:
```
a= 'ram is a good boy'
```

In [35]:
```
a
```

Out[35]:
```
'ram is a good boy'
```

In [36]:
```
1 type(a)
```

Out[36]: str

In [37]:
```
1 b='872492'
```

In [38]:
```
1 b
```

Out[38]: '872492'

In [39]:
```
1 type(b)
```

Out[39]: str

In [40]:
```
1 c='7986987'
```

In [41]:
```
1 type(c)
```

Out[41]: str

In [42]:
```
1 a= 'ram is a good boy'
```

In [ ]:
```
1
```