Python Notes 1

Introduction to Python

Before learning about Python programming language, let us understand what is a programming language and how it works.

An ordered set of instructions to be executed by a computer to carry out a specific task is called a program, and the language used to specify this set of instructions to the computer is called a programming language.

Computers understand the language of 0s and 1s, called machine language or low-level language. High-level programming languages like Python, C++, Visual Basic, PHP, Java are easier for humans to manage.

Python uses an interpreter to convert its instructions into machine language. An interpreter processes the program statements one by one, translating and executing them until an error is encountered or the whole program is executed successfully.

Features of Python

- Python is a high-level, free, and open-source language.
- It is an interpreted language, executed by an interpreter.
- Python programs are easy to understand with a clearly defined syntax.
- Python is case-sensitive.
- Python is portable and platform-independent.
- Python has a rich library of predefined functions.
- Python is helpful in web development.
- Python uses indentation for blocks and nested blocks.

Working with Python

To write and run (execute) a Python program, we need to have a Python interpreter installed on our computer or use any online Python interpreter. (This content is applicable when we work in Python IDLE)

Downloading Python

The latest version of Python 3 is available on the official website: Python.org

Execution Modes

There are two ways to use the Python interpreter: 1. Interactive mode 2. Script mode

1. Interactive Mode

To work in the interactive mode, type a Python statement on the >>> prompt directly. The interpreter executes the statement and displays the results.

2. Script Mode

In the script mode, write a Python program in a file, save it, and then use the interpreter to execute it. Python scripts are saved as files with the `.py` extension.

Python Keywords

Keywords are reserved words with specific meanings to the Python interpreter. Python is casesensitive, so keywords must be written exactly as defined.

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

Identifiers

Identifiers are names used to identify a variable, function, or other entities in a program. Rules for naming an identifier in Python are:

- The name should begin with an uppercase or a lowercase alphabet or an underscore _.
- It can be of any length.
- It should not be a keyword or reserved word.
- Special symbols like !, @, #, $, %, etc., are not allowed.

Example:

```
[2]: # Valid identifiers
marks1, marks2, marks3, avg

# Invalid identifiers
1marks, mark$2, avg#
```

```
  Cell In[2], line 5
    1marks, mark$2, avg#
    ^
SyntaxError: invalid decimal literal
```

Variables

A variable in Python refers to an object that is stored in memory. Variable declaration is implicit, meaning variables are automatically declared when assigned a value.

Example:

```
[ ]: # Variable assignment
length = 10
breadth = 20
area = length * breadth
print(area)   # Output: 200
```

Comments

Comments are used to add a remark or a note in the source code. They are not executed by the interpreter. In Python, a comment starts with `#`.

Example:

```
[ ]: # This is a comment
     amount = 3400   # Variable amount is the total spending on grocery
```

Everything is an Object

Python treats every value or data item as an object, meaning it can be assigned to some variable or passed to a function as an argument.
Example:

```
[ ]: num1 = 20
     print(id(num1))   # Identity of num1
     num2 = 30 - 10
     print(id(num2))   # Identity of num2 and num1 are the same as both refer to
       ↪object 20
```

Data Types

Every value belongs to a specific data type in Python. Data type identifies the type of data values a variable can hold and the operations that can be performed.

Numeric Data Types

Type/Class	Description	Examples
int	Integer numbers	–12, –3, 0, 125, 2
float	Real or floating point numbers	–2.04, 4.0, 14.23
complex	Complex numbers	3 + 4j, 2 – 2j

Boolean Data Type

Boolean data type consists of two constants: True and False. Boolean True value is non-zero, non-null, and non-empty. Boolean False is the value zero.

Example:

```
[ ]: num1 = 10
     print(type(num1))  # <class 'int'>

     var1 = True
     print(type(var1))  # <class 'bool'>

     float1 = -1921.9
     print(type(float1))  # <class 'float'>

     var2 = -3+7.2j
     print(type(var2))  # <class 'complex'>
```

Sequence Data Types

A Python sequence is an ordered collection of items, where each item is indexed by an integer.

- **String:** A group of characters enclosed in single or double quotation marks.

```
[ ]: str1 = 'Hello Friend'
     str2 = "452"
```

- **List:** A sequence of items separated by commas and enclosed in square brackets `[]`.

```
[ ]: list1 = [5, 3.4, "New Delhi", "20C", 45]
     print(list1)
```

- **Tuple:** A sequence of items separated by commas and enclosed in parentheses `()`.

```
[ ]: tuple1 = (10, 20, "Apple", 3.4, 'a')
```

Set Data Type

Set is an unordered collection of items separated by commas and enclosed in curly brackets `{}`.

```
[ ]: set1 = {10, 20, 3.14, "New Delhi"}
```

Dictionary

Dictionary in Python holds data items in key-value pairs. Items in a dictionary are enclosed in curly brackets { }. Dictionaries permit faster access to data. Every key is separated from its value using a colon (:) sign. The key: value pairs of a dictionary can be accessed using the key. The keys are usually strings and their values can be any data type. In order to access any value in the dictionary, we have to specify its key in square brackets [ ].

```
[ ]: dict1 = {'Fruit':'Apple','Climate':'Cold', 'Price(kg)':120}
     print(dict1['Price(kg)'])
```

Operators

Operators are special symbols in Python that carry out arithmetic or logical computation. The value that the operator operates on is called the operand.

Arithmetic Operators

Operator	Description	Example
+	Addition	x + y
-	Subtraction	x - y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

Example:

[4]:
```
x = 15
y = 4

print(x + y)   # 19
print(x - y)   # 11
print(x * y)   # 60
print(x / y)   # 3.75
print(x % y)   #
print(x ** y)   # 50625
print(x // y)   #
```

```
19
11
60
3.75
3
5062
5
3
```

Comparison Operators

Operator	Description	Example
==	Equal to	x == y
!=	Not equal to	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Example:

[3]:
```python
x = 10
y = 12

print(x == y)  # False
print(x != y)  # True
print(x > y)   # False
print(x < y)   # True
print(x >= y)  # False
print(x <= y)  # True
```

```
False
True
Fals
e
True
Fals
e
True
```

Logical Operators

Operator	Description	Example
and	True if both operands are true	x and y
or	True if either operand is true	x or y
not	True if operand is false	not x

Example:

[ ]:
```python
x = True
y = False

print(x and y)  # False
print(x or y)   # True
print(not x)    # False
```

Assignment Operators

Operator	Description	Example
=	Assigns value of right side to left side	x = 5

Operator	Description	Example
+=	Adds right side to left side and assigns to left	x += 5
-=	Subtracts right side from left side and assigns to left	x -= 5
*=	Multiplies left side by right side and assigns to left	x *= 5
/=	Divides left side by right side and assigns to left	x /= 5

%=	Takes modulus using two operands and assigns to left	x %= 5
//=	Floor division on operands and assigns to left	x //= 5
**=	Exponent on operands and assigns to left	x **= 5

Example:

```
[ ]: x = 5

     x += 3  # x = x + 3
     print(x)  #

     x -= 2  # x = x - 2
     print(x)  #

     x *= 4  # x = x * 4
     print(x)  # 24

     x /= 3  # x = x / 3
     print(x)  # 8.0

     \x %= 5  # x = x % 5
     print(x)  # 3.0
```

Identity Operators

Operator	Description	Example
is	Evaluates True if the variables on either side of the operator point towards the same memory location and False otherwise.var1 is var2 results to True if id(var1) is equal to id(var2)	num1 is num2

Operator	Description	Example
is not	Evaluates to False if the variables on either side of the operator point to the same emory location and True otherwise. var1 is not var2 results to True if id(var1) is not equal to id(var2)	num1 is not num2

Example:

[ ]:
```
num1 = 10

print(id(num1))
num2 = num1
print(id(num2))
print(num1 is num2)
print(num1 is not num2)
```

Membership Operator

Operator	Description	Example
in	Returns True if the variable/value is found in the specified sequence and False otherwise	lst = [1,2,3,4], 1 in lst
not in	Returns True if the variable/value is not found in the specified sequence and False otherwise	2 not in lst

Example:

[ ]:
```
lst = [1,2,3,4]

print( 1 in lst)
print(2 not in lst)
```

Expressions

Expressions are combinations of values, variables, operators, and function calls that are interpreted and evaluated by the Python interpreter to produce a result.

Example:

[ ]:
```
x = 5
y = 10

if x < y:
    print("x is less than y ")
else:
    print("x is not less than y ")
```

Statements

A statement is an instruction that the Python interpreter can execute. We have already seen the assignment statement. Some other types of statements that we'll see include `if` statements, `while` statements, and `for` statements.

Example:

[ ]:
```python
x = 5
y = 10

if x < y:
    print("x is less than y ")
else:
    print("x is not less than y ")
```

Input and Output

Interacting with the end use to get input to produce desired result with input statement. In python, We have input() function to get input from the user. This function always returns a string value. To print output, we use print() function.

Example:

[ ]:
```python
your_name = input("Enter your name ")
your_age = int(input("Enter your age "))

print("Hi! ",your_name," and your age is  ", your_age)
```

Type Conversion

Type conversion refers to the conversion of one data type to another. There are two types of type conversion: implicit and explicit.

Example:

[ ]:
```python
# Implicit type conversion
x = 10
y = 2.5
z = x + y
print(z)   # 12.5

# Explicit type conversion
x = 5
y = "10"
z = x + int(y)
print(z)   # 15(z)   # 15
```

Debugging

Debugging is the process of finding and fixing errors in the code.

Example:

[ ]:
```
# Syntax error

10 = x
print(x)
```

[ ]:
```
# runtime error

x = 10
y = 0
print(x/y)
```

[ ]:
```
# logical error
x = 10
y = 20
if x>y:
    print("x is less than y ")
else:
    print("y is less than x ")
```

Assignment

1. Write a program to enter two integers and perform all arithmetic operations on them.

[ ]:

2. Write a program to swap two numbers using a third variable.

[ ]:

3. Write a program to swap two numbers without using a third variable.

[ ]:

4. Write a program to find the average of three numbers. 5. The volume of a sphere with radius r is

$$\frac{4}{(3\pi r^3)}$$

. Write a Python program to find the volume of spheres with radius 7cm, 12cm, 16cm, respectively.

[ ]:

6. Write a program that asks the user to enter their name and age. Print a message addressed to the user that tells the user the year in which they will turn 100 years old.

[ ]: