20 May

Python Basic - 1

Notebook Link- <https://colab.research.google.com/drive/1wAGuJXv9h1X76xjSMIl5RIl9fONxXHw3?usp=sharing>

Q.1. What are keywords in python? Using the keyword library, print all the python keywords.

Ans- Keywords in Python are reserved words that cannot be used as ordinary identifiers. They are used to define the syntax and structure of the Python language.

Keywords are immutable and case-sensitive. There are 35 keywords in python 3.9.

```

import keyword

print(keyword.kwlist)

```

Q.2. What are the rules to create variables in python?

Ans- Rules to declare the variables-

1. **Syntax**: Identifiers can be a combination of letters in lowercase (a to z) or uppercase (A to Z) or digits (0 to 9) or an underscore (\_).
2. **No digits:** They must start with a letter or the underscore character, but not with a digit.
3. **Case-Sensitive**: Identifiers in Python are case-sensitive. For example, myVariable and myvariable are two different identifiers in Python.
4. **No Special Characters**: Identifiers cannot have special characters such as !, @, #, $, %, etc.
5. **Reserved Words**: Python keywords cannot be used as identifiers. Words like for, while, break, continue, in, elif, else, import, from, pass, return, etc. are reserved words. You can view all keywords in your current version by typing help("keywords") in the Python interpreter.
6. **Unlimited Length**: Python does not put any restriction on the length of the identifier. However, it's recommended to keep it within a reasonable size, to maintain readability and simplicity in the code.
7. **Private Identifiers**: In Python, if the identifier starts with a single underscore, it indicates that it is a non-public part of the class, module, or function. This is just a convention and Python doesn't enforce it. If it starts with two underscores, it's a strongly private identifier. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.
8. **Non-ASCII Identifiers**: Python 3 allows the use of non-ASCII letters in the identifiers. This means you can use letters like é, ñ, ö, я, etc. in your identifiers if you wish.

Q.3. What are the standards and conventions followed for the nomenclature of variables in

python to improve code readability and maintainability?

Ans- Python variable naming conventions: Use lowercase with underscores (snake\_case) for variable names, choose descriptive names, avoid reserved words, be consistent, use meaningful names, avoid single leading underscores, use uppercase for constants, and balance clarity and brevity.

Q.4. What will happen if a keyword is used as a variable name?

Ans- It creates a SyntaxError.

if="Keyword"

if

File ["<ipython-input-2-de1a6a82b8a5>"](https://localhost:8080/#), line 1 if="Keyword" ^ SyntaxError: invalid syntax

Q.5. For what purpose def keyword is used?

Ans- The def keyword, followed by the function name, parentheses (), and a colon :. Inside the parentheses, you can specify any parameters your function will take. The function's code block is indented under the function definition.

**User Defined Functions** is a block of reusable code that performs a specific task

```

def myFunction():

pass

```

Q.6. What is the operation of this special character ‘\’?

Ans- the backslash character () is used as an escape character. It is used to indicate that the character immediately following it has a special meaning.

Some common uses of the backslash in Python are:

Escape sequences: The backslash is used to represent special characters within strings. For example, \n represents a newline character, \t represents a tab character, and \" represents a double quote within a string.

```

print("Hello \n world \t !!")

```

Line continuation: The backslash can be used to continue a line of code to the next line. It allows you to break long lines into multiple lines for improved readability.

my\_variable = 10 \

+ 20 \

+ 30

my\_variable

Unicode characters: The backslash can be used to represent Unicode characters using their hexadecimal or octal representation. For example, \u263A represents a Unicode smiley face (☺).

print('Hi \u263A !!')

Q.7. Give an example of the following conditions:

(i) Homogeneous list

(ii) Heterogeneous set

(iii) Homogeneous tuple

Ans- i) Homogeneous list:

A homogeneous list in Python is a list that contains elements of the same data type. Here's an example of a homogeneous list of integers:

(ii) Heterogeneous set:

A heterogeneous set in Python is a set that can contain elements of different data types. Here's an example of a heterogeneous set:

(iii) Homogeneous tuple:

A homogeneous tuple in Python is a tuple that contains elements of the same data type. Here's an example of a homogeneous tuple of strings:

numbers = [1, 2, 3, 4, 5]

my\_set = {1, "hello", 3.14, True}

fruits = ("apple", "banana", "orange", "mango")

print(numbers,my\_set,fruits)

# [1, 2, 3, 4, 5] {1, 3.14, 'hello'} ('apple', 'banana', 'orange', #'mango')

Q.8. Explain the mutable and immutable data types with proper explanation & examples.

Ans- Immutable data types, such as numbers, strings, and tuples, cannot be modified once they are created. Any attempt to change their values results in the creation of a new object. For example, when modifying a string, a new string is created instead of modifying the original one. This immutability ensures the integrity of the data and allows for predictable behavior in the code.

On the other hand, mutable data types, including lists, sets, and dictionaries, can be modified after creation. Changes made to mutable objects directly affect their contents without creating new objects. This allows for dynamic modifications and flexibility in manipulating data structures. However, it requires caution to ensure data integrity and avoid unintended side effects when multiple variables reference the same mutable object.

my\_tuple=(1,2,3,4,5)

my\_tuple[2]=9

my\_tuple

TypeError Traceback (most recent call last)

[<ipython-input-10-0a306e2a505a>](https://localhost:8080/#) in <cell line: 2>()

**1** my\_tuple=(1,2,3,4,5)

----> 2 my\_tuple[2]=9

**3** my\_tuple

TypeError: 'tuple' object does not support item assignment

Q.9. Write a code to create the given structure using only for loop.

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

Ans-

for i in range(1,10,2):

for j in range(i):

print("\*",end='')

print("\n")

Q.10. Write a code to create the given structure using while loop.

|||||||||

|||||||

|||||

|||

|

Ans-

for i in range(10,0,-2):

for j in range(1,i):

print("|",end='')

print("\n")