**Topics:-**

**Background of Python**

**Types of Python packages and differences**

**Exceution way of Python**

**Python keywords**

**Identifiers**

**Variables**

**Overview of Data Types in python**

**List background**

**(1) What is Python and why we should learn this?**

- Easy to learn

- English syntax

- Among the top 5 programming languages since last few years

- Used in Automation

- Used in Data analysis part (COVID 19 status)

- Big Communities (pypi) modules

<https://pypi.org/search/?q=netmiko>

<https://pypi.org/project/netmiko/>

**(2) Types of Python**

Python2 and Python3

Latest: 3.8.3 - some new features/ libraries in built

Python2.x last support date was Jan 1 2020. No new libaries will be on python2

**Main Differences:-**

**print 'Hello, Geeks'** # Python 3.x doesn't support (desinged for only python2.x)

**print('Hope You like these facts')** # Python3 support

raw**\_**input() will return the **string** in python2

input() in python3 in integer

As we can see, if we use parentheses in python 2.x then there is no issue but if we don’t use parentheses in python 3.x, we get SyntaxError.

**(3) Can be execute in two ways:-**

(a) type python on command prompt - **python ide – debug purpose**

(b) Write code and save it with **\*.py extension**

**Basics of Python**

**(1) Python keywords**

Keywords are special words which are reserved and have a specific meaning. Python has a set of keywords that cannot be used as variables in programs.

import keyword

keyword.kwlist

**['and', 'as', 'assert', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'exec', 'finally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'not', 'or', 'pass', 'print', 'raise', 'return', 'try', 'while', 'with', 'yield']**

you cant use this as avariable

Ex:1

>>> for = 1

File "<stdin>", line 1

**for = 1**

^

SyntaxError: invalid syntax

>>> in = 1

File "<stdin>", line 1

in = 1

^

SyntaxError: invalid syntax

>>> in1 = 1

>>> in1

1

>>>

**(2) Identifiers**

(a)It is a name given to variable, class, function.

(b)No space are allowed. (a-z,A-Z,0-9,\_)

(c)They cannot start with numerical **1var is invalid and var1 is valid**

(d)All the combinations including **a-z,A-Z,0-9 or \_** are valid.

for e.g:- myClass and my\_class,var\_1 are logical

(e) Keywords cannot be used as identifiers

**global = 1**

**for = 1**

(f) No special characters are allowed

**a@ = 0 invalid**

(g)case sensitive, **var and VAR** are not same

**(3) Python Variables**

a variable can hold objects of different types.

Ex:1

>>> test = 10

>>> type(test)

<class 'int'>

>>> test = 'techbeamers'

>>> type(test)

<class 'str'>

**>>> h=123**

**>>> h=123**

**>>> type(h)**

**<class 'int'>**

**>>> h1 = str(h)**

**>>> h1**

**'123'**

**>>> type(h1)**

**<class 'str'**>

>>>

>>> h1 = str(h)

>>> h1

'123'

>>> type(h1)

<class 'str'>

>>> h=123456

>>> h1=str(h)

>>> h1

'123456'

>>>

>>>

>>>

>>> h1='adssd'

>>> h1=str(123)

>>> h1

'123'

>>> h1='123'

>>> h1

'123'

>>>

##########################################

Ex:2:-

#!/usr/bin/python

counter = 100 # An integer assignment

miles = 1000.0 # A floating point

name = "John" # A string

print counter

print miles

print name

######################################  
>>> username='admin'

>>>

>>> password= 'admin'

>>>

>>>

>>> ipaddress = '0.0.0.0'

>>>

>>>

>>>

>>> print(username, password, ipaddress)

admin admin 0.0.0.0

>>>

>>>

>>>

################

>>> a=1

>>>

>>>

>>>

>>> a='hello'

>>>

>>>

>>>

>>> a = 1.2

>>>

>>>

>>>

>>> type(a)

<class 'float'>

>>>

>>>

>>> print(a)

1.2

>>>

#################################

ex:3:-

a,b,c = 1,2,"john"

a = 1

b = 2

c = ‘john’

>>> def = 1

File "<stdin>", line 1

def = 1

^

SyntaxError: invalid syntax

>>> def@ = 1

File "<stdin>", line 1

def@ = 1

^

SyntaxError: invalid syntax

>>> def 12121212331 = 1

File "<stdin>", line 1

def 12121212331 = 1

^

SyntaxError: invalid syntax

>>> def1212312 = 1

>>> def1212312

1

>>>

QUICK WRAP:- Python Keywords, Identifiers & Variables.

**Keywords**:- Name which are reserved in Python and cannot used as variables:- True/False

**Identifiers**:- Certain rules to be followed as they are used to identify objects. Like variable\_1 = 10

**Variables**:- It holds the value and its type. It is dynamic

################################################################################

**(4) Python Statements**

**Indentation**:

Many of the high-level programming languages like C, C++, C# use braces { } to mark a block of code. Python does it via indentation.

>>> def hello\_world():

... print("HI. I am new code")

...

>>> hello\_world()

HI. I am new code

>>>

#include <stdio.h>

int main() {

// printf() displays the string inside quotation

printf("Hello, World!");

return 0;

}

**(5) Python Data Types**

What type of data it can store

**Numeric Data types**:-

# Integer number

num = 100

print(num)

print("Data Type of variable num is", type(num))

**Float Data Types**(number with point values in between)

fnum = 34.45

print(fnum)

print("Data Type of variable fnum is", type(fnum))

**String Data Types**

Sequence of the characters a-z

>>> s= 'Hello world'

>>> type(s)

<type 'str'>

>>>

**List Data Types:**

If you have more than one values, then store it in the list.

List1=[1,2,3,4,5]

list2 =[1,3,’abc’,4.5]

Creating an empty list

list1 = []

It doesnot depends on the types of the values

**Dict Data Types**

IF you have more than one values, but in key,values pairs (username, password).

And you want that values very fast. No keys which are duplicates are allowed.

dict1 = {

‘username’ : ‘admin’,

‘password’ : ‘admin

}

Creating empty dictionary

dict1 = {}

Comment in the Python

Single line comment: starts with #

You might prefer to use a single line Python comment when there is need of short, quick comments for debugging. Single-line comments begin with a pound (#) symbol and automatically ends with an EOL (end of the line).

Ex:1

# Define a list of months

months = ['Jan', 'Feb', 'Mar', 'Apr', 'May', 'Jun', 'Jul','Aug','Sep','Oct','Nov','Dec']

Multi line comment: Should in between “”” and “””.

**Operations with List, Dict,Tuple**

(1) List

It can hold values of any types.

List1=[1,2,3,’asdasdas’,’5555’]

- Each values has some position numbers (index). It is used to locate the particular values in the list.

- Position number starts with 0,1,2 and so on.

**Creating a list**

we can create the list with the empty values and with values also,

# Syntax

L1 = [] # An empty list

L2 = [a1, a2,...] # With elements

The list can take any number of elements, and each may belong to a different type (a [number](https://www.techbeamers.com/python-numbers/) or a [string](https://www.techbeamers.com/python-strings-functions-and-examples/), etc.).

# blank list

L1 = []

# list of integers

L2 = [10, 20, 30]

# List of heterogenous data types

L3 = [1, "Hello", 3.4]

- Length of the list

len(list1)

**- Multiple assignment**

>>> init\_list = [0]\*3

>>> print(init\_list)

[0, 0, 0]

**- Multi dimension list also,**

>>> two\_dim\_list = [ [0]\*3 ] \*3

>>> print(two\_dim\_list)

[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

**- Get list elements – list indexing**

- Position number starts with 0,1,2 and so on.

List1=[1,2,3,4,5]

print(list1[0])

1

list1[4]

5

list1[6]

Error: Out of list index

>>> print(two\_dim\_list)

[[0, 0, 0], [0, 0, 0], [0, 0, 0]]

>>> two\_dim\_list[0][2] = 1

Ex: 2

vowels = ['a','e','i','o','u']

consonants = ['b', 'c', 'd', 'f', 'g', 'h', 'j', 'k', 'l', 'm', 'n', 'p', 'q', 'r', 's', 't', 'v', 'w', 'x', 'y', 'z']

#Accessing list elements using the index operator

print(vowels[0])

print(vowels[2])

print(vowels[4])

#Testing exception if the index is of float type

try:

vowels[1.0]

except Exception as ex:

print("Note:", ex)

#Accessing elements from the nested list

alphabets = [vowels, consonants]

print(alphabets[0][2])

print(alphabets[1][2])

OUTPUT:-

a

i

u

Note: list indices must be integers or slices, not float

i

d

#### Reverse indexing

Python enables reverse (Negative) indexing for the sequence data type

it starts with -1

vowels = ['a','e','i','o','u']

print(vowels[-1])

print(vowels[-3])

o/p:-

u

i

**- Extenting the list.**

List1=[1,2,3]

list2 = [4,5,6]

Add values with the help of ‘+’ operator

>>> list1 = [1,2,3]

>>> list2=[2,3,4,5]

>>> list1+list2

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

>>>

### List slicing

**It includes index number only. Not values**

**[start(optional):stop(optional):step(optional)] step = positive (forward direction), negative means reverse direction**

theList = [1, 2, 3, 4, 5, 6, 7, 8]

you want certain index number only

>>> theList[2:5]

[3, 4, 5]

>>> list1=[1,2,3,4,5,6,7]

>>> list1[1:5]

[2, 3, 4, 5]

>>> list1[1:5:1]

[2, 3, 4, 5]

>>>

it will includes, starts with 2,3,4 and will stop before 5(it will excludes)

>>> list1[5:1:-1]

[6, 5, 4, 3]

>>>

- reverse the list

list1[::-1]

>>> list1[::-1]

[7, 6, 5, 4, 3, 2, 1]

>>>

Wants all the values after 2 index number

list1[2:]

Wants all the value before last value

list1[:-1] where : means all

Want to reverse the list

- adding values inthe list

list1.append(‘4’)

>>> list1.append('444')

>>> list1

[1, 2, 3, 4, 5, 6, 7, '444']

>>>

- assignment operator

>>> list1[1]

2

>>> list1[1] = 'test'

>>> list1

[1, 'test', 3, 4, 5, 6, 7, '444']

>>>

- Insert at specific position

>>> list1

[1, 'test', 3, 4, 5, 6, 7, '444']

>>> list1.insert(1,'adasdasdas')

>>> list1

[1, 'adasdasdas', 'test', 3, 4, 5, 6, 7, '444']

>>>

Remove an element from the list

>>> list1

[1, 'adasdasdas', 'test', 3, 4, 5, 6, 7, '444']

>>> list1.remove('adasdasdas') # with the values removal (no index number known)

>>> list1

[1, 'test', 3, 4, 5, 6, 7, '444']

>>> del list1[1] # index number removal

>>> list1

[1, 3, 4, 5, 6, 7, '444']

>>>

Get the index values

>>> list1.index(5)

3

>>> list1

[1, 3, 4, 5, 6, 7, '444']

>>>

- sorted the list

>>> list1

[1, 3, 4, 5, 6, 7, '444']

>>> list1.sort()

>>> list1

[1, 3, 4, 5, 6, 7, '444']

>>>

https://www.techbeamers.com/python-tutorial-step-by-step/#tutorial-list

https://beginnersbook.com/2019/03/python-data-types/

https://www.programiz.com/python-programming/keyword-list