

## Python Notes

- Py high level, object oriented scripted language.
- Py open-source language.
- Platform independent.

### Py Features.

- Py interpreted language- each & every line executes.  
Compiler- will execute al the line once.
- Py interactive- so many command line can directly interact.
- Py is beginner language- easy to use, more inbuilt libraries.
- Dynamically typed- no need to specify the data type explicitly.

### Py history.

- Pt introduced in 1991 by guido van Rosson.

### Execution type.

- Py exe file (comment prompt ale work madadu )& windows IDLE (integrated development learning environment).

### Py used.

- Web development, testing, Data Analysis, D science, AI, ML

### Py (comments & Keywords).

- **Comments-** C are used to write description to unstd logic easily.  
C are non-executable code.  
Py ignore comments in run time.  
Single line comment & multi line comment(#"/" "/")--SL & (""" """)---ML.

#### • **Key-words-**

- In Python, **keywords** are special reserved words that have a fixed meaning in the language.
  - ◊ You **cannot use them as variable names or function names.**
  - ◊ They are used to write the structure (syntax) of Python programs.
-  **Key Points**
- ✓ Keywords are **case-sensitive**
  - ✓ They are **all in lowercase** except True, False, and None
  - ✓ They **cannot be used as identifiers (variable names).**

**Ex:** this is wrong.

if = 10

#### **List of Python Keywords (currently 35)**

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

## **Variable, Data type, Concatenation.**

- **Variable-** V are reserved memory location to store the values.  
V are used to store the data.  
Memory allocated when values are stored in variables.  
Every variables must have some type.

Ex:

A=10 or a=10

Print(type(A)) ----->to check data type.

Print(type(A),A) ----->along with the variable.

Ex:

a,b,name=10,20,"manga"

Ex:

a=b=c=10-----> same value to all.

- **Data types.**

Number, string, list, tuples, dictionary, Boolean.

**Number---** int, float.

Ex:

X=100—int

Y=10.1---float.

String---"welcome".

Boolean—True/False. [ here T & F will be capital].

List-----[1,2,3, 1,"welcomw",10.2,True]

Tuple-----(1,"welcomw",10.2,True) any datatype.

Dictionary----{key:value pari in flower bracket} {1:"manga",2:"neene"}

## **Concatenation.**

- Can concatenate only the same datatype.
- Can do int+float.

Ex:

Print(10+20)

Print(10.5+10.5)

Print("welcome"+ "python")

Print(10+10.5)

Print(True+10)

Print(False+10)

Print(True+True)

---

```
# print(10+20)
# print(10.5+10.5)
# print("welcome"+ "python")
# print("welcome" + " " "python")
# print(10+10.5)
# print(True+10)
```

```
# print(False+10)
# print(True+True)
```

```
# o/p
# 30
# 21.0
# welcomepython
# welcome python
# 20.5
# 11
# 10
# 2
```

#### **# wrong Ex are**

```
# -----
# print(10+"py")
```

```
# o/p
# print(10+"py")
#      ~~~~^~~~~~
# TypeError: unsupported operand type(s) for +: 'int' and 'str'
# -----
```

#### **# wrong Ex are**

```
# -----
# print(True+"welcome")
# # o/p
# print(True+"welcome")
#      ~~~~^~~~~~~~~~~
# TypeError: unsupported operand type(s) for +: 'bool' and 'str'
# -----
```

#### **# Swapping variables.**

```
# a=10
# b=20
# print("before swapping",a,b)
# a,b=b,a
# print("after swapping",a,b)
# -----
```

#### **# redeclaration- declaration can be done many times.**

```
# Ex:
# a=100
# print(a)
# a=101
# print(a)
```

```
# o/p
# 100
```

```
# 101
#
# deleting variables.

# a=10
# print(10)

# b=20
# print(b)
```

```
# del print(a) # deleting the variable- a
# if we call the deleted variable its gives error
```

# Ex:

```
# del print(a)
#      ^^^^^^
# SyntaxError: cannot delete function call
```

---

**# how to take user input.**

```
# input()---function
# raw-input()----take only string data.
```

# Ex:

```
# num1=input("enter 1st number: ")
# num2=input("enter 2nd number: ")
# print(num1+num2)
```

# o/p

```
# enter 1st number: 10
# enter 2nd number: 20
# 1020
# here by default it will take as a input so o/p is 1020. for that we need to type conversion.
# -----
```

**# Type conversion.**

```
# num1=int(input("enter 1st number: "))
# num2=int(input("enter 2nd number: "))
# print(num1+num2)
```

```
## o/p:
# enter 1st number: 10
# enter 2nd number: 20
# 30 # this o/p is after type conversion.
# -----
```

```
# num1=float(input("enter 1st number: "))
```

```
# num2=int(input("enter 2nd number: "))
# print(num1+num2)

# o/p
# enter 1st number: 10.1
# enter 2nd number: 20
# 30.1
# -------

# num1=float(input("enter 1st number: "))
# num2=int(input("enter 2nd number: "))
# print(num1+num2)

# o/p
# here int to float cant possible.
# num2=int(input("enter 2nd number: "))
# ValueError: invalid literal for int() with base 10: '10.5'
# -------

# float can hold integer.
# int can't hold the float.
# -----
```

### # Formatting o/p.

#### # Formatting o/p with % and { }

# type 1

```
name="john"
age=20
sal=15000.50
# print(name,age,sal)
```

# type 2

```
# print("name is: ",name)
# print("sal is: ",sal)
# print("age is: ",age)
```

```
# o/p
# name is: john
# sal is: 15000.5
# age is: 20
```

# type 3

```
# print("name:%s age:%d sal:%f" %(name,age,sal))
# o/p
# name:john age:20 sal:15000.500000
```

# type 4

```
# using { } operator
# print("name:{} age:{} sal:{}".format(name,age,sal))
# o/p
# name:john age:20 sal:15000.5
```

# type 5

```
# index (starts with 0)and value is IMP here
print("name:{0} age:{1} sal:{2}".format(name,age,sal)) # name index 0, age index 1, sal index 2
# name:john age:20 sal:15000.5
# name:john age:john sal:john # here if we change the index values also changes..
# name:20 age:20 sal:20 # here if we change the index values also changes..
# -----
```

## # COndition statements.

### 2 types---- if & elif

**Ex:**

```
a=20
if a>21:
    print("grater:")
else:
    print("not grater:")
```

or.

```
a=10
if 1:
    print("True condition: ")
else:
    print("false condition: ")
```

or.

```
a=10
if True:
    print("True condition: ")
else:
    print("false condition: ")
```

---

## **to check even or odd**

```
a=10
if a%2==0:
    print("even nor: ")
else:
    print("odd nor: ")
```

```
b=9
if b%2==0:
    print("even nor: ")
else:
    print("odd nor: ")
```

---

**statements in single line,  
no need of flower bracket.**

```
print("welcome") if True else print("dont come")
print(" ")
print("welcome") if False else print("dont come")
```

```
print("grater") if 10>5 else print("not grater")
print(" ")
print("grater") if 10>20 else print("not grater")
```

**Multiple statements in single line need flower bracket.**

```
{print("welcm1"),print("mange1")} if True else {print("dont come"),print("manga2")}
print(" ")
{print("welcm1"),print("mange1")} if False else {print("dont come"),print("manga2")}
```

---

```
a=25
if a==10:
    print("correct")
elif a==20:
    print("not correct")
elif a==50:
    print("grt lost")
elif a==90:
    print("gone")
else:
    print(" dam mamar")
```

---

### **take i/p from user**

```
a=int(input("enter a: "))
if a==10:
    print("correct")
elif a==20:
    print("not correct")
elif a==50:
    print("grt lost")
elif a==90:
    print("gone")
else:
    print(" dam mamar")
```

---

### **continue & break**

```
for i in range(1,20):
```

```
    if i%2==0:
        if i==12:
            continue
        print(i)
```

**o/p**

2  
4  
6  
8  
10  
14  
16  
18

```
for i in range(1,20):
```

```
    if i%2==0:
        if i==12:
            break
        print(i)
```

**o/p**

2  
4  
6  
8  
10

---

## **iterative and loops (both r same)**

2 types of loop.  
for & while loop.

### **for loop**

for loop always used for range function.  
range( starts with 0(zero)).

**Ex:**

```
print(list(range(1,10)))
start,End # S,E
o/p
[1, 2, 3, 4, 5, 6, 7, 8, 9]
```

**Ex:**

```
print(list(range(0,11,2)))
start,End,Step # S,E,S
end is excluded in nature
o/p
[0, 2, 4, 6, 8, 10]
```

### **Decreasing order**

**Ex:**

```
print(tuple(range(11,0,-1)))
o/p
(11, 10, 9, 8, 7, 6, 5, 4, 3, 2, 1)
```

**Ex:**

```
print(list(range(-10,-5)))
o/p
[-10, -9, -8, -7, -6]
```

**Ex:**

```
print(list(range(-10,-1,)))
o/p
[-10, -9, -8, -7, -6, -5, -4, -3, -2]
```

### **Ex: negative incremented by 2**

```
print(list(range(-10,-0,2)))
o/p
[-10, -8, -6, -4, -2]
```

---

**Ex:**

**print 1st 10 natural number using for loop**

for i in range(0,11):

    print(i,end=" ")

o/p

0 1 2 3 4 5 6 7 8 9 10

**Ex:**

**print 1st 10 even number using for loop**

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

    print(i,end=" ")

o/p

0 2 4 6 8

**Ex:**

**print 1st 10 odd number using for loop**

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

    print(i,end=" ")

o/p

1 3 5 7 9

**Ex:**

**print 1st 10 natural number using for loop in reverse order**

for i in range(10,0,-1):

    print(i,end=" ")

o/p

10 9 8 7 6 5 4 3 2 1

---

## **While Loop.**

can't use range() function in while loop.

when we know the start value & dont know the ending value then use while loop.

initialization is required & condition is required, increment is required.

**Ex:**

i=1

while i<=5:

    print("manga",i)

    i=i+1

or.

    i+=1

```
o/p  
manga 1  
manga 2  
manga 3  
manga 4  
manga 5
```

**Ex:**

```
i=10  
while i>=1:  
    print(" mama",i)  
    i=i-1
```

```
o/p  
mama 10  
mama 9  
mama 8  
mama 7  
mama 6  
mama 5  
mama 4  
mama 3  
mama 2  
mama 1
```

---

### **Break & continue statements.**

**break:** when want to come out from the loop based on some condition.

**Ex:**

```
for i in range(1,10):  
    if i==5:  
        break  
    print(i)
```

```
o/p  
1  
2  
3  
4
```

**continue-**

**Ex:**

```
for i in range(1,10):  
    if i==5:  
        continue
```

```
print(i,end=" ")
```

**o/p**  
1 2 3 4 6 7 8 9

---

### **Numbers:**

numbers are two type----int and decimal/float.

```
x=10 #----int  
y=10.1 #----float  
print(type(x))  
print(type(y))
```

**o/p**  
<class 'int'>  
<class 'float'>

---

### **Type conversion:**

#### **Ex:**

```
x=10 #int  
print(float(x))  
print(type(float(x)))
```

**o/p**  
10.0  
<class 'float'>

#### **Ex:**

```
y=10.0  
print(type(y))  
print(int(y))  
print(type(int(y)))
```

**Ex:**  
<class 'float'>  
10  
<class 'int'>

---

### **Built in functions in python.**

2 built in functions----Max() & Min()

#### **Ex:Max()**

```
large=max(10,20,30,40,50,60)
print(large)
```

o/p  
60

### **Ex:Min()**

```
small=min(10,20,30,12,10,5,47,22)
print(small)
```

o/p  
5

---

## **STRINGS.**

### **Properties of string.**

str support both homogeneous & heterogeneous collection.

str support duplicate character.

str are ordered collection.

str support both +ve & -ve index.

str support slicing operation.

indexing can access only character.

slicing can access multiple values.

str is a collection of character enclosed in double or single quotes.

### **creating strings.**

```
name="manga"
name2='manga2'
name3="""manga3"""
```

```
print(name)
print(name2)
print(name3)
```

or.

```
name1=str("mamu")
print(name1)
```

str are immutable in python.(once created cant be modified)  
every object store in one memory, we use id() to get address.

### **Ex:**

```
name1=str("mamu")
print(name1)
print(id(name1))
```

o/p  
mamu  
1980542730032

---

### **operations on string.**

in str support indexing, & index starts with 0.StopAsyncIteration  
use + and \* to perform operations on string.

+ is used to concatenate.

\* used to repeating same things multiple times.

**Ex:**

```
str="welcome"
print(str + "to python")
```

```
str="hi"
print(str*3)
```

### **Str Slicing.**

extractin the sub string from the original str by using [ ].

we have to give starting & ending index

syntax: str[start:end]

**Ex:**

```
str="welcome"
print(str[0:4])
print(str[0])
print(str[1])
print(str[2])
print(str[3])
print(str[4])
print(str[5])
print(str[6])
```

o/p  
welc  
w  
e  
l  
c  
o  
m

```
e
str="welcome"
print(str[0:])
# welcome
print(str[:7])
welcome
```

ending index excluded

---

ord()=return the ASCII code of the character.  
chr()= return character represented by a ASCII nor.

**Ex:**

```
print(ord('A'))
print(chr(65))
```

---

### **str functions in python.**

len()=return the length of the str.  
max()=return character having highest ASCII value.(jasti)  
min()=return character having lowest ASCII vlaue.(kammi)

```
str="welcome"
print(len(str))
print(max(str))
print(min(str))
# o/p
7
w
c
```

---

### **in & notin operator.**

in & notin operator are used to check existence of string in another string.  
they also known as membership operator.

**Ex: for in**

```
str="welcom"
print("wel" in str)
o/p
True
print("dl" in str)
o/p
False
```

**Ex: for notin**

```
str="welcom"
print("del" not in str)
o/p
True
print("wel" not in str)
o/p
False
```

---

### Iterating string using for loop

**Ex:**  
s="python"  
for i in s:  
 print(i,end=" ") #---- here we r printing i, it will give characters in the s.  
o/p  
p y t h o n  
 print(s,end=" ") #---- here we r printing s, it will print the characters in the s depending on the length.  
o/p  
python python python python python python

**Ex:**  
str="mamu"  
for i in str:  
 print(i,end=" ") # m a m u

or.

Ex: using range  
s="mamu"  
for i in range(len(s)):  
 print(s[i],end=" ") # m a m u

---

### string indexing.

```
str="mamuma"
print(str[0]) # m
print(str[0:4]) # mamu
print(str[:6]) # mamuma
print(str[0:]) # mamuma
```

---

### string reversing using for loop

```
str="mamuma"
for i in range(len(str)-1,-1,-1): # S, E, S
    print(str[i],end=" ") # a m u m a m
```

```
print(str[::-1],end=" ") # amumam amumam amumam amumam amumam amumam
```

### reverse user input.

```
user=input("enter name: ")
reverse=""
for i in user:
    reverse=i+reverse
print(reverse)
```

o/p

```
enter name: mamama
```

```
m
```

```
am
```

```
mam
```

```
amam
```

```
mamam
```

```
amamam
```

---

### str concatenation

```
first_name="ma"
last_name="nga"
full_name=first_name + last_name
print(full_name)
```

### repetition of a str

```
str="manga "
print(str*3)
o/p
manga manga manga
```

---

### str methods()

upper()= converting str to upper.

lower()=converting str to lower.

strip()=removes leading & trailing space from a string.

replace(old,new)=replace substring with another string.

```
str=" welcome to python "
print(str.upper())
print(str.lower())
print(str.strip()) # here it will remove the spaces, see o/p for clarity
print(str.replace("python","snakepython"))
o/p
```

## WELCOME TO PYTHON

```
welcome to python  
welcome to python  
welcome to snakepython
```

---

### accessing str characters

```
str="learn please"
```

```
print(str[0])
```

```
print(str[1])
```

```
print(str[2])
```

```
print(str[3])
```

```
o/p
```

```
|
```

```
e
```

```
a
```

```
r
```

```
print(str[:3]) # lea
```

```
print(str[1:]) # earn please
```

```
print(str[:8]) # learn pl
```

```
print(str[2:6]) # arn
```

start, end, step

step ge 2 kotray 1 skip agate.

step ge 2 kotray 2 skip agate

position starts with 1.

index starts with 0.

---

### escape sequence

```
s="manga \nis a big manga." # here \n is refere to new line IMP
```

```
print(s)
```

```
# o/p
```

```
# manga
```

```
# is a big manga.
```

```
s="manga \tis a big manga." # here \t is refere to tap space. IMP
```

```
print(s)
```

```
# o/p
```

```
# manga  is a big manga
```

## Comprehension.

In Python, comprehension is a short and fast way to create a collection (list, set, dict).

✓ The data types that support comprehension:

Data Type	Comprehension Name	Example
List	List Comprehension	[x for x in range(5)]
Set	Set Comprehension	{x for x in range(5)}
Dictionary	Dictionary Comprehension	{x: x*x for x in range(5)}

---

### ❖ Examples

#### 1 List Comprehension

```
squares = [x*x for x in range(1, 6)]
print(squares)
```

#### 2 Set Comprehension

```
numbers = {x for x in range(1, 6)}
print(numbers)
```

#### 3 Dictionary Comprehension

```
d = {x: x*x for x in range(1, 6)}
print(d)
```

---

## X String & Tuple do NOT support comprehension directly

You cannot do:

`tuple(x for x in range(5)) X`

But you can convert a generator to tuple:

```
t = tuple(x for x in range(5)) # ✓ valid conversion
```

---

### ✿ Final Answer

❖ Only List, Set, and Dictionary support comprehension directly.

Sure! Here are simple 2 examples for each comprehension (list, set, and dictionary) in easy English 

---

### ❖ List Comprehension

Used to create a list in one line.

✓ Example 1: Squares of numbers 1 to 5

```
squares = [x*x for x in range(1, 6)]
```

```
print(squares)
```

Output: [1, 4, 9, 16, 25]

✓ Example 2: Even numbers from 1 to 10

```
evens = [x for x in range(1, 11) if x % 2 == 0]
```

```
print(evens)
```

Output: [2, 4, 6, 8, 10]

---

#### ❖ Set Comprehension

Used to create a set (no duplicates).

✓ Example 1: Multiples of 3 from 1 to 10

```
multiples = {x for x in range(1, 11) if x % 3 == 0}
```

```
print(multiples)
```

Output: {3, 6, 9}

✓ Example 2: Characters in a word (unique only)

```
letters = {ch for ch in "programming"}
```

```
print(letters)
```

Output (order may change): {'p', 'r', 'o', 'g', 'a', 'm', 'i', 'n'}

---

#### ❖ Dictionary Comprehension

Used to create a dict (key-value pairs) quickly.

✓ Example 1: Number and its square

```
square_dict = {x: x*x for x in range(1, 6)}
```

```
print(square_dict)
```

Output: {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

✓ Example 2: Convert lowercase to uppercase

```
upper = {ch: ch.upper() for ch in "abc"}
```

```
print(upper)
```

Output: {'a': 'A', 'b': 'B', 'c': 'C'}

---

### ➊ Summary

Type	Brackets	Example
List Comprehension	[]	[x for x in range()]
Set Comprehension	{}	{x for x in range()}
Dictionary Comprehension {key:value} {x:x*x for x in range()}		

### Strings. [CGPT]

Sure 

Here is a simple and clear explanation of STRING in Python.

#### ❖ What is a String in Python?

A string is a sequence of characters written inside quotes.

Examples:

```
name = "Bharath"
```

```
msg = 'Hello World'
```

```
data = "12345"
```

❖ A string can contain letters, numbers, symbols, and spaces.

---

#### ❖ How to write a string

Type	Example
------	---------

Type	Example
Single quotes	'hello'
Double quotes	"hello"
Triple quotes (multi-line)	'''hello''' or """hello"""
Example of multi-line string:	
	text = """This is multi-line string""" print(text)

---

#### ◊ Accessing characters in a string

We can access characters using index numbers.

```
s = "python"
print(s[0]) # p
print(s[3]) # h
```

❖ Index starts from 0.

---

#### ◊ String Slicing

Extract part of a string.

```
s = "python"
print(s[0:3]) # pyt
print(s[2:6]) # thon
```

---

#### ◊ Important String Functions

Function	Meaning
len(s)	length of string
s.upper()	convert to uppercase
s.lower()	convert to lowercase
s.title()	first letter caps
s.strip()	remove spaces
s.replace(a, b)	replace text
s.split()	convert string to list

#### Example:

```
s = " hello world "
print(len(s))
print(s.upper())
print(s.strip())
print(s.replace("world", "Python"))
```

---

#### ◊ Looping through a string

```
for ch in "hello":
    print(ch)
```

---

## ◊ Joining (Concatenation)

```
a = "Hello"  
b = "Python"  
print(a + " " + b)
```

---

## ◊ Strings are immutable

❖ You cannot change characters in a string directly.

✗ Not allowed:

```
s = "python"  
s[0] = 'P'  
✓ Correct way (create new string):  
s = "python"  
s = "P" + s[1:]  
print(s)
```

---

## قطرة Small practice programs

□ Reverse a string

```
s = input("Enter string: ")  
print(s[::-1])
```

□ Count vowels in a string

```
s = input("Enter string: ")  
count = 0  
for ch in s:  
    if ch in "aeiouAEIOU":  
        count += 1  
print("Vowels:", count)
```

□ Check if string is palindrome

```
s = input("Enter string: ")  
if s == s[::-1]:  
    print("Palindrome")  
else:  
    print("Not Palindrome")
```

## interview question on string for freshers with answer.

ChatGPT said:

Sure 🤗

Here are the top interview questions on STRING for freshers with simple answers.

---

## ◊ Basic Questions

1. What is a string in Python?

A string is a sequence of characters written inside quotes.

Example:

```
name = "Python"
```

---

2. How do you create a string in Python?

By using:

- Single quotes ''
  - Double quotes " "
  - Triple quotes """ "" or """ """
- 

### 3. Are strings mutable or immutable in Python?

Strings are immutable, meaning you cannot change characters directly.

Example:

```
s = "abc"
```

```
s[0] = "x" # X Not allowed
```

---

### 4. How to find length of a string?

Use len()

```
s = "hello"
```

```
print(len(s)) # 5
```

---

### 5. How to access a single character from a string?

Using indexing.

```
s = "python"
```

```
print(s[0]) # p
```

```
print(s[3]) # h
```

---

◊ Intermediate Questions

### 6. What is string slicing?

Extracting part of a string using index range.

```
s = "python"
```

```
print(s[1:4]) # yth
```

---

### 7. How to convert a string to uppercase and lowercase?

```
s = "Hello"
```

```
print(s.upper()) # HELLO
```

```
print(s.lower()) # hello
```

---

### 8. How to remove spaces from beginning and end of a string?

```
s = " hello "
```

```
print(s.strip()) # "hello"
```

---

### 9. How do you replace text in a string?

```
s = "hello world"
```

```
print(s.replace("world", "Python")) # hello Python
```

---

### 10. How to split a string?

Splits string into list.

```
s = "a,b,c"
```

```
print(s.split(",")) # ['a','b','c']
```

---

◊ Program-Based Questions

11. Reverse a string

```
s = "python"  
print(s[::-1]) # nohtyp
```

---

12. Count vowels in a string

```
s = "hello"  
count = 0  
for ch in s:  
    if ch in "aeiouAEIOU":  
        count += 1  
print(count)
```

---

13. Check if string is palindrome (same front & back)

```
s = "madam"  
if s == s[::-1]:  
    print("Palindrome")  
else:  
    print("Not Palindrome")
```

---

14. Count number of words in a string

```
s = "I love Python"  
print(len(s.split()))
```

---

15. Remove all spaces from a string

```
s = "I love Python"  
print(s.replace(" ", ""))
```

---

◊ Advanced Questions for Freshers

16. Difference between == and is in strings

Operator Meaning

==      Compares values

is      Compares memory address

---

17. How to check if string contains only alphabets?

```
s = "Hello"  
print(s.isalpha()) # True
```

---

18. How to check if string contains only digits?

```
s = "12345"  
print(s.isdigit()) # True
```

---

19. How to join list items into a string?

```
items = ["a", "b", "c"]  
print(",".join(items)) # a,b,c
```

---

## 20. Difference between strip(), lstrip(), and rstrip()

### Function Meaning

strip()	Removes spaces both sides
lstrip()	Removes left spaces
rstrip()	Removes right spaces

---

**Below is the explanation in very simple English with two different examples (one for for loop and one for while loop).**

---

#### ◊ FOR LOOP — Simple English

❖ A for loop is used when you know how many times you want to repeat the code.

👉 Example 1: Print numbers 1 to 5

```
for i in range(1, 6):
```

```
    print(i)
```

⌚ Meaning: Start from 1, go up to 5, print each number one by one.

👉 Example 2: Print each letter in a word

```
for ch in "hello":
```

```
    print(ch)
```

⌚ Meaning: Take each letter from the word hello and print it.

---

#### ◊ WHILE LOOP — Simple English

❖ A while loop is used when you do not know how many times the code will repeat and you want to repeat until a condition becomes false.

👉 Example 1: Print numbers 1 to 5

```
i = 1
```

```
while i <= 5:
```

```
    print(i)
```

```
    i = i + 1
```

⌚ Meaning: Keep printing numbers while i is less than or equal to 5.

👉 Example 2: Keep asking password until correct

```
password = ""
```

```
while password != "1234":
```

```
    password = input("Enter password: ")
```

```
print("Login success")
```

⌚ Meaning: Continue asking for password again and again until the correct one is entered.

---

### ● Summary in 1 line

for loop              When number of repetitions is known

while loop            When number of repetitions is not known and depends on condition

⌚ Basic Interview Questions

- 
- 1 What is a loop in Python?
  - 2 Difference between for loop and while loop.
  - 3 When do you choose for loop instead of while loop?
  - 4 What is an iteration?
  - 5 What is an infinite loop? Give example.
  - 6 Can for loop work without range()?
  - 7 What happens if we forget to update the condition variable in a while loop?
- 

#### Program-Based Questions

- 8 Write a program to print numbers from 1 to 10 using a for loop.
  - 9 Write a program to print numbers from 1 to 10 using a while loop.
  - 10 Print even numbers from 1 to 50.
  - 11 Print multiplication table of a number entered by user.
  - 12 Find the sum of first n natural numbers using loop.
  - 13 Print factorial of a number using loop.
- 

#### Concept-Based Questions

- 14 What does the range() function do in a loop?
- 15 Can range() go in reverse? Example?
- 16 Explain break, continue, and pass statements inside loops.
- 17 Difference between i++ in Java/C and i += 1 in Python.
- 18 What will happen if break is used inside a loop?
- 19 What will happen if continue is used inside a loop?
- 20 Can we use else with a loop? If yes, why?

Sure 

Here are the interview questions with answers in very simple English.

---

#### Basic Interview Questions & Answers

##### 1. What is a loop in Python?

A loop is used to repeat a block of code multiple times.

---

##### 2. Difference between for loop and while loop

For Loop

While Loop

Used when we know how many times to repeat	Used when we don't know how many times to repeat
--	--

Works with range(), lists, strings, etc. Runs based on a condition

---

##### 3. When do you choose for loop instead of while loop?

Use for loop when the number of repetitions is fixed or countable (like print 1–10).

---

##### 4. What is an iteration?

One cycle or one repetition of the loop.

---

## 5. What is an infinite loop?

A loop that never stops because the condition always remains true.

---

## 6. Can for loop work without range()?

Yes. It can loop over:

- list
- string
- tuple

Example:

```
for ch in "hello":  
    print(ch)
```

---

## 7. What happens if we forget to update the condition variable in a while loop?

The loop never stops → it becomes an infinite loop.

---

### ◊ Program-Based Questions

## 8. Print numbers 1 to 10 using for loop

```
for i in range(1, 11):  
    print(i)
```

---

## 9. Print numbers 1 to 10 using while loop

```
i = 1  
while i <= 10:  
    print(i)  
    i = i + 1
```

---

## 10. Print even numbers from 1 to 50

```
for i in range(2, 51, 2):  
    print(i)
```

---

## 11. Print multiplication table of a number

```
n = int(input("Enter number: "))  
for i in range(1, 11):  
    print(n, "x", i, "=", n * i)
```

---

## 12. Sum of first n natural numbers

```
n = int(input("Enter n: "))  
sum = 0  
for i in range(1, n + 1):  
    sum = sum + i  
print(sum)
```

---

## 13. Factorial of a number

```
n = int(input("Enter n: "))  
fact = 1
```

```
for i in range(1, n + 1):
    fact = fact * i
print(fact)
```

---

◊ Concept-Based Questions & Answers

14. What does range() do?

It creates a sequence of numbers for looping.

---

15. Can range() go in reverse?

Yes.

```
for i in range(10, 0, -1):
    print(i)
```

---

16. Explain break, continue and pass.

Keyword Meaning

break      Stops the loop

continue    Skips current iteration, continues next

pass       Does nothing (placeholder)

---

17. Difference between i++ in Java/C and i += 1 in Python

Python does not support i++.

Instead, we write:

i += 1

---

18. What happens if break is used inside a loop?

Loop stops immediately.

---

19. What happens if continue is used inside a loop?

Skips current repetition and goes to the next.

---

20. Can we use else with loop?

Yes. else runs only when loop completes normally (without break).

---

◊ Tricky Questions with Answers

21. Output?

```
for i in range(3):
    print("Hello")
```

Output:

Hello

Hello

Hello

---

22. Output?

i = 1

```
while i < 5:
```

```
print(i)
Infinite loop because i never increases.
```

---

23. Output?  
for i in range(1, 10, 2):  
 print(i)

Output:

```
1
3
5
7
9
```

---

24. Output?  
i = 1  
while i == 1:  
 print("Hi")  
Infinite loop because condition always true.

---

◊ Advanced Questions & Answers

25. Is for loop faster than while loop?  
In Python, both are almost same speed.  
Choice depends on logic, not speed.

---

26. Can we use loops inside loops?  
Yes. It is called nested loops.

---

27. What is looping through a list or dictionary?

Repeating code for each item:

```
for x in [10, 20, 30]:  
    print(x)
```

---

28. What are iterables and iterators?

Term      Meaning

Iterable A collection we can loop over (list, string, tuple)

Iterator Object that gives values one by one

---

29. What is the role of loops in pattern printing?  
Loops print shapes using stars (\*) like triangle, square pattern, etc.

---

30. How to exit from nested loops?  
Using break inside the inner loop.