27-09-2024

Training Day- 9

IDENTIFIERS VS VALUES:

IDENTIFIERS: User defined names: Set of rules:

Using Combination Of Following Characters:
1. English alphabets: A to Z, a to z
2. Numbers or : 0 to 9
3. underscore, : _

We Create Identifiers
Special symbols other than underscore not allowed, should not start
with numbers to create identifiers. keywords can't be identifiers
Identifiers are created directly without using quotes or any
special symbol
Python is a case sensitive langauge

EXAMPLES OF IDENTIFIERS: Variable name, function name or class name
Values: Different type of values: Data Types
…….New Program…….
```
# print(100)
# print(2.35)
# print(100,200,300)
```
…….New Program…….
```
# x=100
# temp=200
# flag123=500
# cetpa_2345=900
```
…….New Program…….
```
# a*b=200        #SyntaxError
# a b=500 #Error
# 1ab=82  #Identifier names, can't start with numbers
```
…….New Program…….
```
# a=100
# A=200
# print(a,A)
```
…….New Program…….
```
# Ram=5
# print(ram)      #NameError:
```

IDENTIFIER: Collection of particular characters:
User defined names 1. Variable 2. Functions 3. Classes

KEYWORDS: Predefined names

if, elif, for, def.....
Values: Different types of values: Data types:
Single element:

int	23, -92
float	2.35, 4.0
complex	2+3j
bool	True, False
NoneType	None

MULTI ELEMENT: Iterators In Python, **multi-element** refers to any data structure or operation that involves multiple elements. This typically applies to collections like lists, tuples, sets, dictionaries, or other iterables where multiple elements can be stored or processed together
str : Collection of characters: using single, double, or triple quotes
list
tuple
dict
set
frozenset
"""

```
# #New Program
# a=b         #NameError:
# print(a)

# #New Program
# a=5
# print(a)
# print(type(a))
```

str: Collection of characters:

"""

.........New Program..........:: Single Line String

```
# a='Welcome to CETPA'
# print(a)
# a="Welcome to CETPA"
# print(a)
# a='''Welcome to CETPA'''
# print(a)
# a="""Welcome to CETPA"""
# print(a)
```

.........New Program..........: Multi Line String

```
# a='''Welcome to CETPA.
# Thanks for joining CETPA'''
# print(a)
# a="""Welcome to CETPA.
```

# Thanks for joining CETPA"""
# print(a)
……….New Program……….:
# a="b"   #"b" is a value of string type
# print(a)
print(Comma separated arguments)
print(values,variable,expressions,conditions,functions,classes)

"""
# a,b=5,7
# s="CETPA"
# cetpa="cetpa"
# print(99,True,None,a,s,a+b,a>b,len(s),type(s))

"""
Escape Characters:
\n   : New Line
\t   : Tab Character

If we want to print quotes in a string:
double quotes print: put string in single quotes
single quotes print: put string in double quotes
or
put the quotes in front of \
\' : Single quotes
\" : Double quotes
"""
……….New Program……….:
# print("A\t\t\tBCD")
# print("A\n\n\nBCD")
……….New Program……….:
# x="CETPA"
# print(x)
……….New Program……….:
# print("he said, \"Hello!\"")
"""
String Is A Collection Of Characters:
'abc----zAB----Z012...9@#$'

"""

……….New Program……….:
'cetpa'

```
# print('"cetpa"')
```

……….New Program……….:

```
"cetpa"
# print('"cetpa"')
```

……….New Program……….:

```
"cetpa"
# print('"ce"tpa"')
```

……….New Program……….:

```
"cetpa"
# print("\'ce\'tpa\'")
```

……….New Program……….:

```
"cetpa"
# print("\'ce\'t"pa\'")

"""
```

Python Supports Unicode Formats: ie in string, you can write world's any language : special symbol, currency symbol, alphabets...

……….New Program……….:

```
# print("का")
```

Optional Arguments: end, and sep variables: str type

```
end="\n"
sep=" "
```

Print Function: sep variable is automatically printed in between the arguments and end variable is automatically printed at the end of the arguments.

```
"""
# #New Program
# a,b,c,d,e=1,2,3,4,5
# print(a,b,c)        #asepbsepcend
# print(d,e)          #dsepeend
# print(a,b,c,sep="*",end="#$")    ##asepbsepcend
```

……….New Program……….: # a,b,c,d,e=1,2,3,4,5

```
# print(a,b,c,sep="*",end="#")
# print(d,e,sep="$")
```

……….New Program……….:

```
# print(sep="*",end="#")
# print()
# print()
```

How to use comments:

```
# use
```

Can make strings in the program
Multiple lines to use # : Shortcut key: cntrl + /
"""
```
# #New Program
# print("CETPA")
# print("Hello")
```

LOGICAL OPERATORS: Logical operators in Python are used to perform logical operations on Boolean values, which are either True or False. They are essential for creating conditional statements and expressions, and for implementing complex decision-making processes. Following the all are logical operator

and #English wala and, hindi wala aur
or #Englsih wala or ya hindi wala ya
not #Wahi nahi
and: if both inputs are True then output is True else False
or: if at least one of the inputs is True then output is True else False
"""

……..New Program………..
```
# a,b=5,7
# print(a==5 and b==7)
# print(a>=5 and b<7)
# print(a>=5 or b<7)
```
……..New Program………..
```
# a,b=5,7
# print(a==b)
# print(a<b)
```
……..New Program………..
```
# id=input("Enter the id:")
# pwd=input("Enter the pwd:")
# if(id=="007" and pwd=="bond"):
#     print("Correct id and pwd")
# else:
#     print("Incorrect id or pwd")
```

"""
Packing and unpacking concept
"""
```
# #New Program
# a,b=5,7
# print(a,b)

# #New Program
# a,b="AB"        #Python supports Unpacking directly
# print(a,b)

# #New Program
# a,b,c,d,e="CETPA"        #Python supports Unpacking directly
# print(a,b,c,d,e)
# #New Program
# a,b,c,d="CETPA"        #Error
# print(a,b,c,d)

# #New Program
# a,b,c,d,e,f="CETPA"        #Error
# print(a,b,c,d,e)

# #New Program
# a="C","E","T","P","A"
# print(a,type(a))

# #New Program
# a=False,False
# print(a)
# print(type(a))
```

"""

Nested Conditions:

Conditions inside conditions

"""

```
# #New Program
# a,b,c=1,2,3
# if(a>=1):
#     print("Inside First If")
#     if(b<=2):
#         print("Inside Second If")
#     elif(b==3):
#         print("Inside First elif")
#     else:
#         print("ABCD")
# elif(a==0):
#     print("BCDE")
#     if(b==1):
#         print("CDEF")

# #New Program
# a=5
# if(a==5):
#     print("CETPA")
# elif(a==5):        #else if
#     print("Hello")
# else:
#     print("ABCD")
```

Comparison of 3 numbers

```
# #Find the bigger between 2 numbers
# no1=input("Enter First No:")
# no2=input("Enter Second No:")
# if(no1>no2):
#     print(no1,"is the bigger no")
# else:
#     print(no2,"is the bigger no")

# #Find the bigger between 2 numbers or check if they are equal
# no1=input("Enter First No:")
# no2=input("Enter Second No:")
# if(no1>no2):
#     print(no1,"is the bigger no")
# elif(no1<no2):
#     print(no2,"is the bigger no")
# else:
#     print("Both numbers are equal")


# #New Program: Find the biggest of 3 numbers: Using nested conditions
# no1=int(input("Enter First No:"))
# no2=int(input("Enter Second No:"))
# no3=int(input("Enter Third No:"))
# if(no1>no2):    #If no2 is not the biggest, we will go inside if
#     if(no1>no3):
#         print(no1,"is the biggest no")
#     else:
#         print(no3,"is the biggest no")
# else:       #If we reach at else, it means no1 is not the biggest no
#     if (no2 > no3):
#         print(no2, "is the biggest no")
#     else:
#         print(no3, "is the biggest no")

# #New Program
# print("1" > "5")
# print("1" < "5")
# print("11" > "5")
# print(11 > 5)
# print("ABC" > "AAC")

"""

"""
```