PYTHON FUNDAMENTALS

Class – XI

2020-21

CHARACTER SET

A set of valid characters recognized by python. Python uses the traditional ASCII character set. The latest version recognizes the Unicode character set.

The ASCII character set is a subset of the Unicode character set

- ❖ Letters :– A-Z, a-z
- ❖ Digits :– 0-9
- ❖ Special symbols :– Special symbol available over keyboard
- ❖ White spaces:– blank space,tab,carriage return,new line, form feed
- ❖ Other characters:- Unicode

INPUT AND OUTPUT

```
var1='Computer Science'
var2='Informatics Practices'
print(var1,' and ',var2,' )
```

Output :-

```
        Computer Science and Informatics Practices
```

e.g.

```
age = int(input('enter your age'))
percentage = float(input('enter percentage'))
```

input() Function In Python allows a user to give input to a program from a keyboard but returns the value accordingly.

e.g.

```
age = int(input('enter your age'))
C = age+2 #will not produce any error
```

NOTE : input() function always enter string value in python 3.so on need int(),float() function can be used for data conversion.

TOKEN

Smallest individual unit in a program is known as token.

1. **Keywords**
2. **Identifiers**
3. **Literals**
4. **Operators**
5. **Punctuators**

KEYWORDS

Reserved word of the compiler/interpreter having specific meaning / functionalitywhich can’t be used as identifier.

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

IDENTIFIERS

A Python identifier is a name used to identify a variable, function, class, module or other object.

- ❖ An identifier starts with a letter A to Z or a to z or an underscore (_) followed by zero or more letters, underscores and digits (0 to 9).
- ❖ Python does not allow special characters
- ❖ Identifier must not be a keyword of Python.
- ❖ Python is a case sensitive programming language.

Thus, **Rollnumber** and rollnumber are two different identifiers in Python.

Some valid identifiers : Mybook, file123, z2td, date_2, _no

Some invalid identifier : 2rno,break,my.book,data-cs

IDENTIFIERS

Some additional naming conventions

1. Class names start with an uppercase letter. All other identifiers start with a lowercase letter.

2. Starting an identifier with a single leading underscore indicates that the identifier is private.

3. Starting an identifier with two leading underscores indicates a strong private identifier.

4. If the identifier also ends with two trailing underscores, the identifier is a language-defined special name.

LITERALS

Literals in Python can be defined as number, text, or other data that represent values to be stored in variables.

Example of String Literals in Python
name = ‘Johni’   ,   fname =“johny”

Example of Integer Literals in Python(numeric literal)
age = 22

Example of Float Literals in Python(numeric literal)
height = 6.2

Example of Special Literals in Python
name = None

LITERALS

Escape sequence

Escape Sequence	Description
\\	Backslash (\)
\'	Single quote (')
\"	Double quote (")
\a	ASCII Bell (BEL)
\b	ASCII Backspace (BS)
\f	ASCII Formfeed (FF)
\n	ASCII Linefeed (LF) (New Line)
\r	ASCII Carriage Return (CR)
\t	ASCII Horizontal Tab (TAB)
\v	ASCII Vertical Tab (VT)
\ooo	Character with octal value ooo
\xhh	Character with hex value hh

OPERATORS

Operators are defined as symbols that are used to perform operations on operands.

Types of Operators

1. Arithmetic Operators.
2. Relational Operators.
3. Assignment Operators.
4. Logical Operators.
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

OPERATORS

1. Arithmetic Operators

Arithmetic Operators are used to perform arithmetic operations like addition, multiplication, division etc.

Operators	Description	Example
+	perform addition of two number	a+b
-	perform subtraction of two number	a-b
/	perform division of two number	a/b
*	perform multiplication of two number	a*b
%	Modulus = returns remainder	a%b
//	Floor Division = remove digits after the decimal point	a//b
**	Exponent = perform raise to power	a**b

OPERATORS

2. Relational Operators
Relational Operators are used to compare the values.

Operators	Description	Example
==	Equal to, return true if a equals to b	a == b
!=	Not equal, return true if a is not equals to b	a != b
>	Greater than, return true if a is greater than b	a > b
>=	Greater than or equal to , return true if a is greater than b or a is equals to b	a >= b
<	Less than, return true if a is less than b	a < b
<=	Less than or equal to , return true if a is less than b or a is equals to b	a <= b

OPERATORS

3. Assignment Operators: Used to assign values to the variables.

Operators	Description	Example
=	Assigns values from right side operands to left side operand	a=b
+=	Add 2 numbers and assigns the result to leftoperand.	a+=b
/=	Divides 2 numbers and assigns the result to left operand.	a/=b
*=	Multiply 2 numbers and assigns the result to leftoperand.	A*=b
-=	Subtracts 2 numbers and assigns the result to leftoperand.	A-=b
%=	modulus 2 numbers and assigns the result to leftoperand.	a%=b
//=	Perform floor division on 2 numbers and assigns the result to leftoperand.	a//=b
**=	calculate power on operators and assigns the result to leftoperand.	a**=b

OPERATORS

4. Logical Operators
Logical Operators are used to perform logical operations on the given two variables or values.

Operators	Description	Example
and	return true if both condition are true	x and y
or	return true if either or both condition are true	x or y
not	reverse the condition	not(a>b)

```
a=30
b=20
if(a==30 and b==20):
   print('hello')
```

Output :-

```
hello
```

OPERATORS

6. Membership Operators

The membership operators in Python are used to validate whether a value is found within a sequence such as such as strings, lists, or tuples.

Operators	Description	Example
in	return true if value exists in the sequence, elsefalse.	a in list
not in	return true if value does not exists in the sequence, elsefalse.	a not in list

E.g.

```
a = 22
list = [22,99,27,31]
In_Ans = a in list
NotIn_Ans = a not in list
print(In_Ans)
print(NotIn_Ans)
```

Output :-

```
True
False
```

OPERATORS

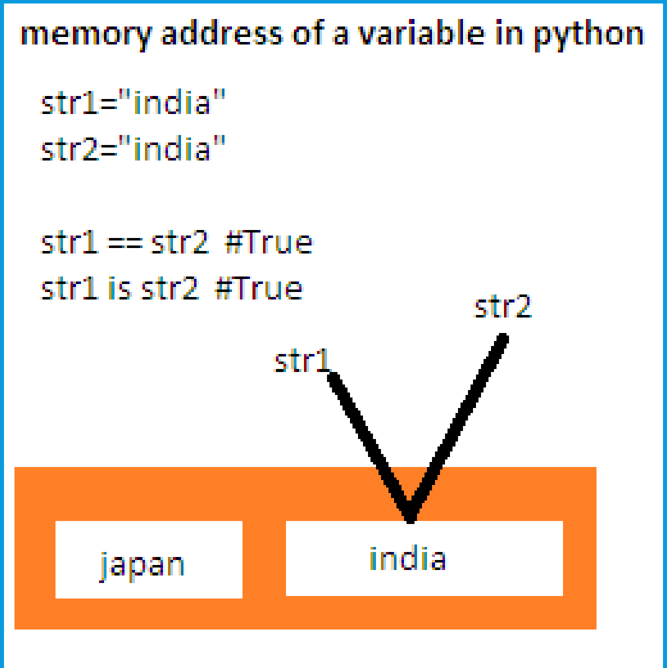
7. Identity Operators :dentity operators in Python compare the memory locations of two objects.

Operators	Description	Example
is	returns true if two variables point the same object, else false	a is b
is not	returns true if two variables point the different object, else false	a is not b

```
e.g.
a = 34
b=34
if (a is b):
    print('both a and b has same identity')
else:
    print('a and b has different identity')
b=99
if (a is b):
    print('both a and b has same identity')
else:
    print('a and b has different identity')
```

Output :-

```
both a and b has same identity
a and b has different identity
```



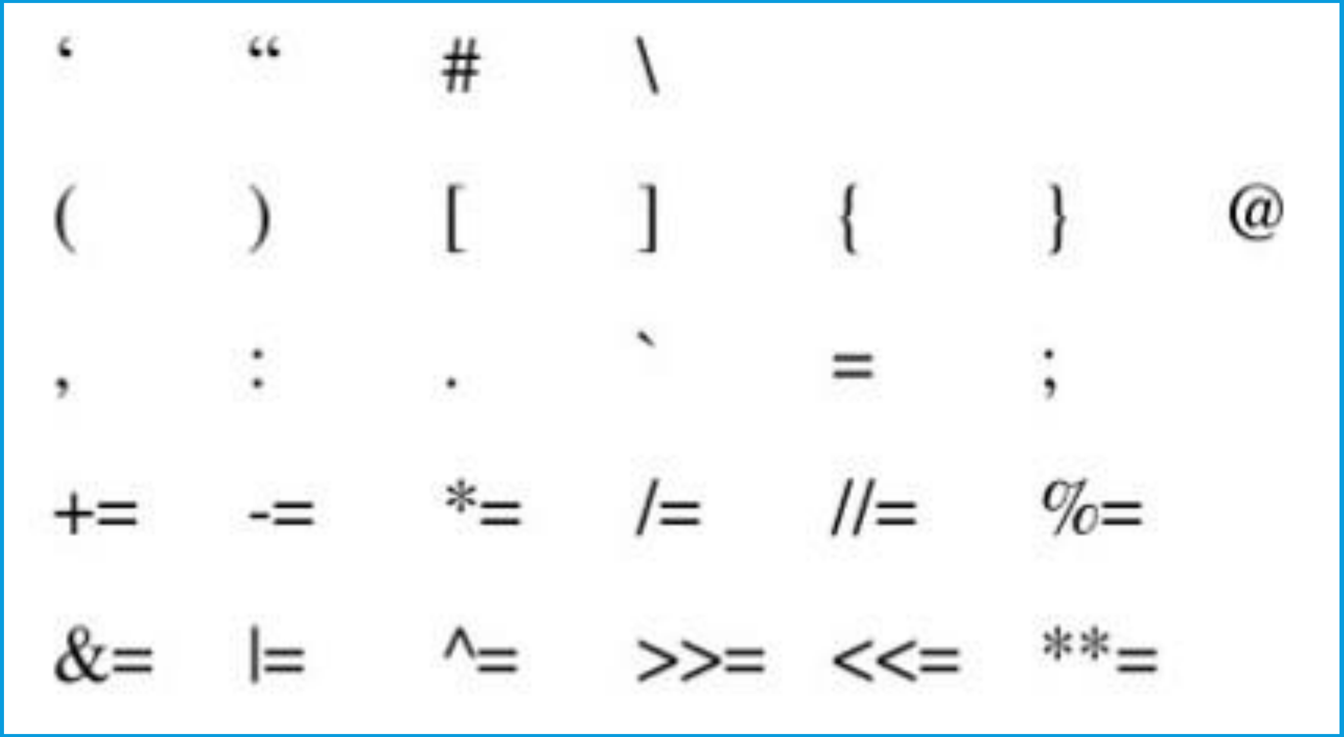
memory address of a variable in python

```
str1="india"
str2="india"

str1 == str2  #True
str1 is str2  #True
```

Punctuators

Used to implement the grammatical and structure of a Syntax. Following are the python punctuators.



‘	“	#	\			
(	)	[	]	{	}	@
,	:	.	`	=	;	
+=	-=	*=	/=	//=	%=	
&=	|=	^=	>>=	<<=	**=	

ASSIGNMENT

- Assignment is already uploaded in the classroom