CHAPTER II : PYTHON FUNDAMENTALS

CLASS XI

COMPUTER SCIENCE

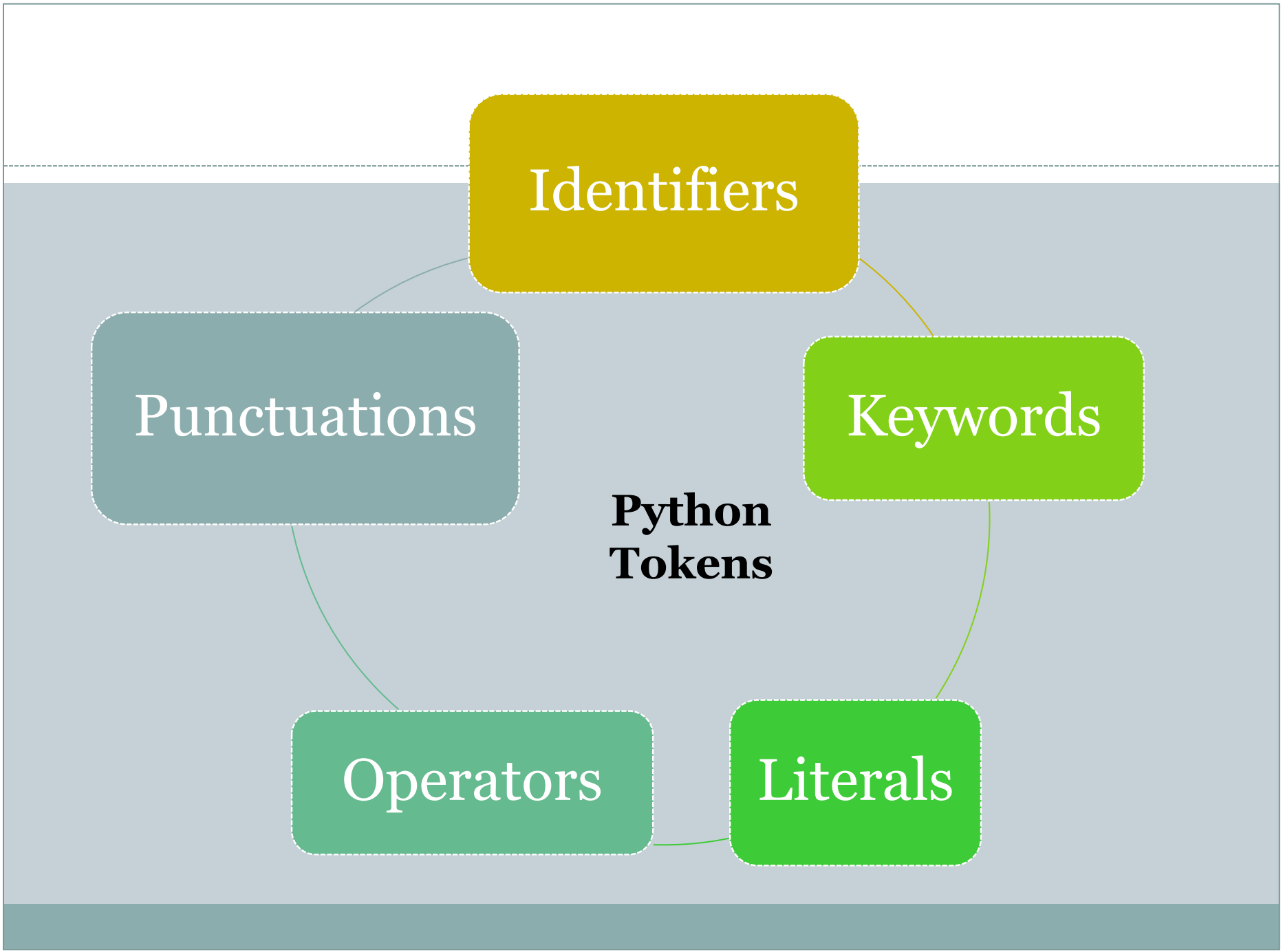
Python Character Set

- Character set is asset of valid characters that a language can recognize. A character can represents any letter, digit, or any other sign. Following are some of the python character set.

LETTERS	A to Z and a to z
DIGITS	0 -9
SPECIAL SYMBOLS	space ,+ -* ^ \ [] {} = != < > . _ ; : & #, under score(_)
WHITE SPACE	Blank space , horizontal tab (– >), carriage return , Newline, Form feed.
OTHER CHARACTERS	Python can process all ASCII and Unicode characters as part of data or literals.

Tokens:

The smallest individual unit in a program is known as token or lexical unit. A token can be any keyword, Identifier, Literals, Punctuators, and Operators.


Identifiers
Punctuations
Keywords
Python
Tokens
Operators
Literals

Identifiers

Identifiers are names given to identify *something*.

Identifiers are fundamental building blocks of a program and are used to give names to different part of the program that is variables, objects, classes, functions, lists, dictionaries etc.

Identifiers (Rules)

There are some rules you have to follow for naming identifiers:

- Must begin with a letter (uppercase ASCII or lowercase ASCII or Unicode character) or an underscore ('_').

- The rest of the identifier name can consist of letters (uppercase ASCII or lowercase ASCII or Unicode character), underscores ('_') or digits (0-9).

- Identifier names are case-sensitive.

  For example, myname and myName are not the same. **Note the lowercase n in the former and the uppercase N in the latter.**

Identifiers (Rules)

- Examples of *valid* identifier names are i, ___my_name, name_23.

  Examples of ”invalid” identifier names are 2 things, this is spaced out, my-name, >a1b2_c3 and “this_is_in_quotes”.

- An identifier must not be a keyword of Python.
- Can be of any length.

Keywords

- Keywords are words having special meaning reserved by Python language.
- They are used by Python interpreter to recognize the structure of program.
- As these words have specific meaning for interpreter, they cannot be used for any other purpose.

Keywords(cont.)

- A partial list of keywords in Python 3.6.x is

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

Literals / Values

- The data items which never change their value throughout the program run. There are several kind of literals:

- String Literals
- Numeric Literals
- Boolean Literals
- Special Literal *None*

Literal (cont.)

- **1. String Literals:** It is a sequence of letters surrounded by either by single or double or triple quotes. g “abc” , ‘a’ , “raman”.

Literals(contd.)

2. Numeric Literals: The numeric literals in Python can belong to any of the following different numerical types:

- **Integer literals (both int and long):** Integer constant (literal) must have at least one digit and must not contain any decimal point. It may contain either (+) or (-) sign. A number with no sign is assumed to be positive. Commas cannot appear in an integer constant.
- For example: 12344, 41, +68, -23

Literals(contd.)

- **Floating Point Literals:** A real constant (literal) in fractional form must have at least one digit before a decimal point and at least one digit after the decimal point. It may also have either + or – sign preceding it. A real constant with no sign is assumed to be positive.
- For example: 2.0 , 18.6 , -23.9 , +89.66

Literal (cont.)

- **Complex number:**
- Complex number in python is made up of two floating point values, one each for real and imaginary part. For accessing different parts of variable (object) x; we will use x.real and x.image. Imaginary part of the number is represented by 'j' instead of 'i', so 1+0j denotes zero imaginary part.
- **For example**
- >>> x = 1+0j
- >>> print x.real,x.imag
- 1.0 0.0
- **Example**
- >>> y = 9-5j
- >>> print y.real, y.imag
- 9.0 -5.0

Literal (cont.)

- **Boolean Literals:** A Boolean literals in Python is used to represent one of the two Boolean values that is **True** or A Boolean literal can either have value as True or as false.

Special Literal *None*

- None is a special type in Python that represents nothingness.
- ***For example, it is used to indicate that a variable has no value if it has a value of None.***
- The None literal is used to indicate something that has not yet been created. It is also used to indicate the end of lists.

Operators

- Operators are special symbols that perform specific operations on one, two, or three operands, and then return a result.

Operators

- **Unary Operator**

Description	Symbol	Example 1	Example 2
unary plus	+	If x=10, then +x means 10	If x=-10, then +x means -10
unary minus	–	If x=10, then -x means -10	If x=-10, then +x means 10

Operators

- **Arithmetic Operator**

		>>>10+5	>>>'Hello'+'Amjad'
Addition			
	+	15	HelloAmjad
		>>>10-5	>>>30-70
Subtraction	–	5	-40
		>>>10*5	>>>'Hello'*3
Multiplication	*	50	HelleoHelloHello
		>>>17/5	>>28/3
		3	9
		>>>17/5.0	>>28/7.0
Division		3.4	4
		>>>17.0/5	
	/	3.4	

Operators

- **Arithmetic Operator**

Remainder / Modulo	%	>>>16%5 1	>>>13%5 3
Exponent	**	>>>2**3 8	>>>16**0.5 4
Floor division (integer division)	//	>>>7.0//2 3	>>>5//2 2

Operators

- **Bitwise Operator**

Operation	Operator	Use	Description
Bitwise AND	&	X & Y	The AND operator compare two bits and generate a result of 1 if both bits are 1; otherwise, it return 0.
Bitwise exclusive OR (XOR)	^	X^Y	The EXCLUSIVE – OR operator compare two bits and returns 1 if either of the bits are 1 and gives 0 if both bits are 0 and 1.
Bitwise OR	|	X | Y	The OR operator compare two bits and generate a result of 1 if both bits are complementary; otherwise, it return 0.
Bitwise Complement	~	~X	The COMPLEMENT operator is used to invert all of the bits of the operands.

Operators

- **Identity operator**

Operation	Operator	Use	Description
Is the identity same?	is	X ix Y	Return True if both its operands are pointing to same object (i.e, both referring to same memory location), returns false otherwise
Is the identity not same?	is not	X is not Y	Return True if both its operands are pointing to different object (i.e, both referring to different memory location), returns false otherwise

Operators

- **Relational Operator**

Less than	<	>>>5<7
		TRUE
		>>>7<5
		FALSE
		>>> 5<7<10
		TRUE
		>>> 5<7 and 7<10
		TRUE
			>>>'Hello' < 'Amjad'
			FALSE
			>>>'Amjad' < 'Hello'
			TRUE

Greater than	<	>>>7>5 TRUE >>>10<10 FALSE	>>>'Hello' > 'Amjad' TRUE >>>'Amjad' > 'Hello' FALSE

Operators

- **Relational Operator**

Less than or equal to	>=	>>> 2<=6
		TRUE
		>>> 6<=4
		FALSE
			<<< 'Hello' <='Amjad'
			FALSE
			>>>'Amjad'<='Hello'
			TRUE

Greater than or equal to	>=	>>> 2>=6
		FALSE
		>>> 6>=4
		TRUE
			<<< 'Hello' >='Amjad'
			TRUE
			>>>'Amjad'>='Hello'
			FALSE

Operators

- **Relational Operator**

Equal to	==	>>> 5==5 TRUE >>>10==11 FALSE	>>>'Hello'=='Hello TRUE >>>'Hello'=='Good Bye' FALSE
Not equal to	!= , < >	>>>10!=11 TRUE >>>10!=10 FALSE	>>>'Hi!='HI' TRUE >>>'HI'!='HI' FALSE

Operators

Assignment Operators

=	Assignment	It is use to assign the value to the variable	a=6	A will become 6
/=	Assign quotient	Divided and assign back the result to left operand	>>> p/=2	p will become 5
+=	Assign sum	Added and assign back the result to left operand	>>> p+=2	p will become 12
*=	Assign product	multiplied and assign back the result to left operand	>>> p*=2	p will become 20

Operators

- **Assignment Operators**

%=	Assign remainder	Taken modulus using two operands and assign the result to left operand	>>> p%=2	p will become 0
-=	Assign difference	subtracted and assign back the result to left operand	>>> p-=2	p will become 8
**=	Assign Exponent	Performed exponential(power) calculation on operators and assign value to the left operand	>>> p**=2	p will become 100
//=	Assign floor division	Performed floor division on operators and assign back the result to left operand	>>> p//=2	p will become 5

Operators

- **Logical Operators**

and	Logical AND	X and Y	If both the operand is true, then the condition becomes **True**
or	Logical OR	X or Y	If any one of the operand is true, then the condition becomes **True**
not	Logical NOT	not X	Reverses the state of the operand/ condition.

Operators

Membership Operators

in	Whether variable in sequence	The **in operator** tests if a given value is contained in a sequence or not and return True or False Accordingly	>>>3 in [1,2,3,4] **TRUE**
not in	Whether variable not in sequence	The **not in operator** tests if a given value is contained in a sequence or not and return True or False Accordingly	>>>6 not in [1,2,3,4,5] **True** (Because the value 6 is not in the sequence.) >>>4 not in [1,2,3,4,5,6] **FALSE**

Punctuators

- The following nine ASCII characters are the separators:
- ( ) { } [ ] ; , . \ # @ : = ‘ “

Variable

- A variable is a labelled storage location whose value can be accessed or changed.

Simple assignment

E.g., movie_name='Python World'

↘ ↘

LHS Value RHS Value

E.g., x=100

y=359

z=x+50

Multiple assignment

E.g., x=y=10

it means x=10 and y=10

E.g., x, y = 12, 50

it means x=12 and y=50

E.g.,

x, y = 12, 50 → python first resolves RHS

x, y = y, x → x, y = 50, 12

print(x,y) → output : 50, 12

Multiple assignment

E.g.,

a, b =2, 3 → a = 2 b = 3

b, a= a+ b, a-b → b = 2+3=5 a=2-3=-1

print(a,b) → -1, 5

e.g.,

```
print(result)
result= 1 + 4
print(result)
```

If we write this code, then output will be:NameError : name ‘result’ is not defined

- We cannot print a variable which is not created.First we have to create a variable ‘result’.
- If we rewrite the code, then output will be 5.

```
result= 1 + 4
print(result)
```

Dynamic Typing

- Python dynamically detects the type of data.
- E.g., student=10 #numeric variable
  print(student)

  student=computer #string variable
  print(student)

We cannot do: student=computer
print(student/2) # it's a string variable, we cannot divide it by 2.

Input & Output

Input + Processing = Output

Syntax: <variable_name>=input(<msg>)

e.g., student_name=input("Enter name :")
print(student_name)

Output: Enter name : Computer Science
Computer Science

Input & Output

- Input method returns value which is by default string.
- This will be detected by using python's built-in method "type".
- The method "type" tells us what is the type of the declared variable.
- E.g., name=science result=244

type(name) type(result)

Output

str int

Input & Output

- E.g., if we want to input age from the user, it will be by default in string format, for this we have convert it from string data to integer data & for that we need int() or float() method.

Syntax: variable_name=int(input(msg))

```
age = int(input("Enter your Age:"))
Print('Age is:', age)
```

Output:

Enter your Age: 20

Age is: 20

Input & Output

- Output using print statement

  print(value, [sep=‘’, end=‘\n’])

“sep‘” is used to separate two values & “end” is used to end lines.