



Programing

Prof. K. Adisesha



Python!

- Created in 1991 by Guido van Rossum
 - Named for Monty Python
- Useful as a **scripting language**
 - **script**: A small program meant for one-time use
 - Targeted towards small to medium sized projects
- Used by:
 - Google, Yahoo!, Youtube
 - Many Linux distributions
 - Games and apps (e.g. Eve Online)





Python

- Interpreted
 - You run the program straight from the source code.
 - Python program → Bytecode → a platform's native language
 - You can just copy over your code to another system and it will automatically work! *with python platform
- Object-Oriented
 - Simple and additionally supports procedural programming
- Extensible – easily import other code
- Embeddable – easily place your code in non-python programs
- Extensive libraries
 - (i.e. reg. expressions, doc generation, CGI, ftp, web browsers, ZIP, WAV, cryptography, etc...) (wxPython, Twisted, Python Imaging library)





Installing Python

Windows:

- Download Python from <http://www.python.org>
- Install Python.
- Run **Idle** from the Start Menu.

Note: For step by step installation instructions, see the course web site.

Mac OS X:

- Python is already installed.
- Open a terminal and run `python` or run Idle from Finder.

Linux:

- Chances are you already have Python installed. To check, run `python` from the terminal.
- If not, install from your distribution's package system.





Programming basics

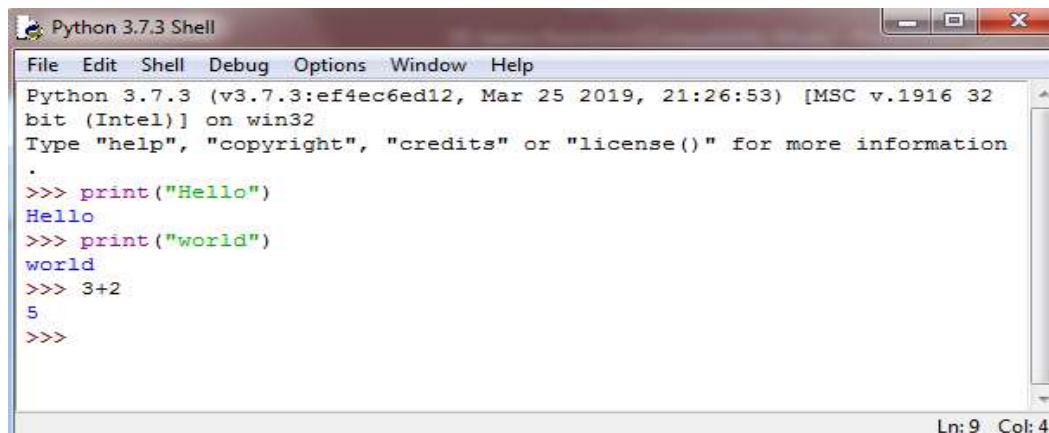
- **code** or **source code**: The sequence of instructions in a program.
- **syntax**: The set of legal structures and commands that can be used in a particular programming language.
- **output**: The messages printed to the user by a program.
- **console**: The text box onto which output is printed.
 - Some source code editors pop up the console as an external window, and others contain their own console window.





The Python Interpreter

- **interpreted**
 - In Python Code is written and then directly executed by an **interpreter**
 - Type commands into interpreter and see immediate results
- Allows you to type commands one-at-a-time and see results
- A great way to explore Python's syntax
 - Repeat previous command: Alt+P



```
Python 3.7.3 Shell
File Edit Shell Debug Options Window Help
Python 3.7.3 (v3.7.3:ef4ec6ed12, Mar 25 2019, 21:26:53) [MSC v.1916 32
bit (Intel)] on win32
Type "help", "copyright", "credits" or "license()" for more information
.
>>> print("Hello")
Hello
>>> print("world")
world
>>> 3+2
5
>>>
```





Token

Smallest individual unit in a program is known as token.

- 1. Keywords
- 2. Identifiers
- 3. Literals
- 4. Operators
- 5. Punctuators / Delimiters





Keywords

- Reserve word of the compiler/interpreter which 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
except		





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**





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`





Operators

- Operators can be 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





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





Relational Operators

- **Relational Operators are used to compare the values.**

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





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 left operand.	a+=b
/=	Divides 2 numbers and assigns the result to left operand.	a/=b
=	Multiply 2 numbers and assigns the result to left operand.	A=b
-=	Subtracts 2 numbers and assigns the result to left operand.	A-=b
%=	modulus 2 numbers and assigns the result to left operand.	a%=b
//=	Perform floor division on 2 numbers and assigns the result to left operand.	a//=b
=	calculate power on operators and assigns the result to left operand.	a=b





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)

Example:

```
a=30
```

```
b=20
```

```
if(a==30andb==20):
```

```
    print('hello')
```

Output :-

```
hello
```





Membership Operators

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

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

- **Example:**

```
a = 22
```

```
list = [22,99,27,31]
```

```
Ans1= a in list
```

```
print(Ans1)
```

```
Ans2= a not in list
```

```
print(Ans2)
```

- **Output :-**

True

False





Identity Operators

- **Identity 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

- **Example:**

```
a = 34
```

```
b=34
```

```
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
```





Punctuators / Delimiters

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

‘ “ # \

() [] { } @

, : . ` = ;

+= -= *= /= //= %=

&= |= ^= >>= <<= **=





Python program

- A python program contain the following components
 1. Comments
 2. Function
 3. Expression
 4. Statement
 5. Block & indentation





Python Comment

- **Comments:** which is readable for programmer but ignored by python interpreter
 - a) Single line comment: Which begins with # sign.
- Syntax:
 - # **comment text (one line)**
 - **Example**
This is a comment
 - b) Multi line comment (doc-string): either write multiple line beginning with # sign or use triple quoted multiple line. E.g.
 - **Example**
"""this is my
first
python multiline comment """





Expressions

- **expression:** A data value or set of operations to compute a value.

Examples: $1 + 4 * 3$
 42

- Arithmetic operators we will use:

+ - * / addition, subtraction/negation, multiplication, division
% modulus, a.k.a. remainder
** exponentiation

- **precedence:** Order in which operations are computed.

- * / % ** have a higher precedence than + -

$1 + 3 * 4$ is 13

- Parentheses can be used to force a certain order of evaluation.

$(1 + 3) * 4$ is 16





Type conversion

- The process of converting the value of one data type (integer, string, float, etc.) to another data type is called type conversion.
- Python has two types of type conversion.
- **Implicit Type Conversion:** Python automatically converts one data type to another data-type. This process doesn't need any user involvement.
- **Explicit Type Conversion:** In Explicit Type Conversion, users convert the data type of an object to required data type. We use the predefined functions like `int()`, `float()`, `str()` etc.





Variables

- **variable:** A named piece of memory that can store a value.
 - Usage:
 - Compute an expression's result,
 - store that result into a variable,
 - and use that variable later in the program.

- **assignment statement:** Stores a value into a variable.
 - Syntax:

name = ***value***

- Examples: $x = \boxed{5}$ $gpa = \boxed{3.14}$

- A variable that has been given a value can be used in expressions.
 $x + 4$ is 9





The print Statement

- Python uses indentation to indicate blocks, instead of { }
- Makes the code simpler and more readable
- In Python, you **must** indent.
- **print** : Produces text output on the console.

- Syntax:

```
print (" Message ")
```

```
print ( Expression )
```

- Prints the given text message or expression value on the console, and moves the cursor down to the next line.

```
print ( Item1, Item2, ..., ItemN )
```

- Prints several messages and/or expressions on the same line.

- Examples:

```
print ("Hello, world!")
```

```
age = 45
```

```
print ("You have", 65 - age, "years until retirement")
```

Output:

```
Hello, world!
```

```
You have 20 years until retirement
```





input

- `input` : Reads a number from user input.
 - You can assign (store) the result of `input` into a variable.
 - Example:

```
age = input("How old are you? ")
print "Your age is", age
print "You have", 65 - age, "years until
retirement"
```

Output:

```
How old are you? 53
Your age is 53
You have 12 years until retirement
```

