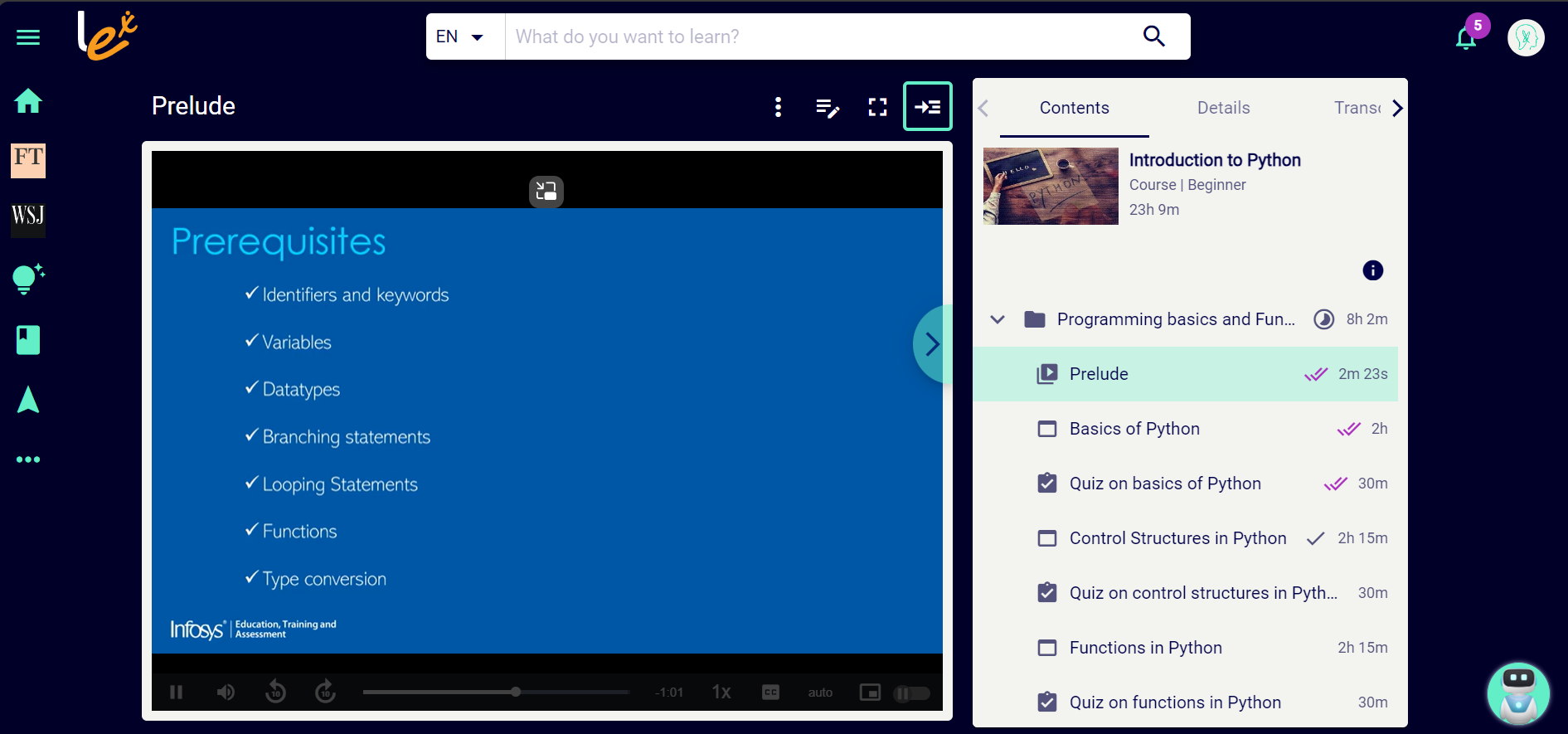
PYTHON NOTES: LEX COURSE : INTRODUCTION TO PYTHON

A screenshot of a computer

Description automatically generated



A screenshot of a computer

Description automatically generated

BASICS OF PYTHON: (8 PAGES )

PAGE 1:

**Following are some of the important features of Python:**

**Python is open source:** The Python implementation is under an open source license that makes it freely usable and distributable, even for commercial use.

**Python is interpreted:** Python is a high-level language which is interpreted by a Python interpreter.

**Python is cross platform compatible:** Python can be executed on all major platforms like Windows, Linux/Unix, OS/2, Mac and others.

**Python is Object-Oriented:**In Python, data is encapsulated within the objects as it supports the object-oriented style of programming.

**Python is a great choice for new learners:** Python is easy to learn and follows a simple syntax, so it is a good choice for beginner programmers. Python also supports wide range of application development.

**Python is extensible:** Python has a wide range of libraries and built-in functions which helps in easy and rapid development of applications.

**Python is interactive:** Python users are provided a command prompt where they can interact directly with the interpreter to write programs.

**Database connectivity:** Python provides interfaces required to connect to all major databases like Oracle, MySQL, PostgreSQL and others.

Next, you will learn about identifiers and keywords in Python.

PAGE 2:

**Identifiers:**

In Python, variables, functions, classes, modules and objects are identified using a name known as an identifier. An identifier can start with an uppercase or lowercase character or an underscore (\_) followed by any number of underscores, letters and digits. All identifiers in Python are case sensitive.

**Example:**weight=10

In the above example, *weight* is an identifier.

**Keywords:**

Keywords are the **reserved words** in Python. So keywords cannot be used as identifiers or to name variables and functions. Few of the keywords are listed below.

**Example:** if, else, elif, for, where, break, continue

Next, you will learn about variables and datatypes in Python.

PAGE 3 :

Variables are like containers for data (i.e. they hold the data) and the value of the variable can vary throughout the program.

**Declaring a variable:**

**Syntax:**var\_name = literal\_value

where var\_name is the name given to the container holding the value specified as literal\_value in the syntax above.

**Example:** weight=10

In the above example, *weight* is the container holding the value *10*  which can change during the execution of the program.

Python may have data belonging to different types. Common data types used in programming are listed below:

| **Category** | **Datatype** | **Example** |
| --- | --- | --- |
| Numeric | int | 123 |
| Numeric with decimal point | float | 123.45 |
| Alphanumeric | string | Hello |
| Boolean | Boolean | True, False |

**Python is a dynamically typed language!**

In the above example, no **datatype** was mentioned at the time of declaring variable. In Python, the datatype of a variable is decided automatically at the time of execution based on the value assigned to it. This is called as **dynamic typing**.

num=65 #line 1

num="A" #line 2

In line 1, variable *num* is assigned a value *65* which is an integer, so the data type of *num* variable is integer in line 1.

In line 2, variable *num* is assigned a value “*A*” which is a string, so the data type of the *num* variable is string in line 1.

**Note:** To check the datatype of the variable you can use **type(var\_name)**which in turn returns the **datatype** of the variable.

**Example:**

1. num=65
2. print(num,type(num))
3. num="A"
4. print(num,type(num))

**Output:**

65 <class 'int'>  
A <class 'str'>

Next, you will learn about input and output statements used in Python.

PAGE 4:

## The input() function:

Python provides the **input()** built-in function to read an input from the user using the standard input device (i.e. keyboard). The input() function always returns string data irrespective of the type of data entered through the keyboard.

**Syntax:**var\_name = input([“interactive statement”])

where,

**var\_name** is the variable assigned with the string value which is read using input method.

**Interactive statement** is the statement displayed to the user expecting the response from them.

**Example:**

1. input\_var=input("please enter the value")
2. print(input\_var)

**sample output:**

please enter the value100  
100

## The print() function:

Python provides the **print()** built-in function to display the output onto the standard output device (i.e. Monitor)

**Syntax:**print(“var\_name1, var\_name2, …”, [end=”value1”, sep=”value2”])

where,

**var\_name1, var\_name2** are the variable names or the literals you want to print or output

**end** is used to specify the separator between two print statements which is ‘\n’ by default

**sep** is used to specify the separator between multiple variables displayed using a single print statement

**Example:**

1. a="infy"
2. b=20.127
3. c=10
4. print(a,b,c)
5. print(a,b,c,sep=":")
6. print(a,b,c,end=" ")
7. print(a,b,c)
8. print("b=%0.2f" %b)
9. print("c=%8d" %c)
10. print("c=%-8d" %c)

**Output:**

infy 20.127 10  
infy:20.127:10 #seperator between variables changed to ‘:’  
infy 20.127 10 infy 20.127 10 #seperator between two print statement changed to " "  
b=20.13          #as the format is 0.2 value is rounded of two decimal digits  
c=      10         #right aligned within the reserved 8 spaces   
c=10               #left aligned within the reserved 8 spaces as there is – symbol

Next, you will see various operators in Python.

PAGE 5:

Operators in Python are the symbols used to indicate the operation to be performed. Some of the most common operators used in Python are listed below:

| **Category** | **Operators** |
| --- | --- |
| Arithmetic Operators | +,-,\*,/, %,// |
| Relational Operators | ==,!=,>,<,>=,<= |
| Assignment Operators | =,+=,-=,\*=,/=,%= |
| Logical Operators | and,or,not |

**Arithmetic operators:**

| **Operator** | **Explanation** | **Example** |
| --- | --- | --- |
| + | Used for addition operation | "+" is used as addition operator where 11+2 is evaluated as 13 |
| - | Used for subtraction operation | "-" is used as subtraction operator where 11-2 is evaluated as 9, 2-11 is evaluated as -9 |
| \* | Used for multiplication operation | "\*" is used as multiplication operator where 11\*2 is evaluated as 22 |
| / | Used for division operation | "/" is used as division operator where 11/2 is evaluated as 5.5 |
| // | Used for integer division operation | "//" is used for integer division where 11//2 is evaluated as 5 |
| % | Used for modulo operation. Consider the expression  num1% num2 which finds the remainder after dividing num1 by num2 | "%" is used as modulo operator where 11%2 is evaluated as 1, 9%11 is evaluated as 9 |

**Relational operators:**

| **Operator** | **Explanation** | **Example** |
| --- | --- | --- |
| == | Used for checking the equality of two values/variables | 10 == 10 is evaluated as True  10 == 100 is evaluated as False |
| != | Used for checking the inequality of two values/variables | 10 != 10 is evaluated as False  10 != 100 is evaluated as True |
| > | Used for checking if num1 is greater than num2 and is represented as num1 > num2 | 10 > 10 is evaluated as False  100 > 10 is evaluated as True |
| < | Used for checking if num1 is lesser than num2 and is represented as num1 < num2 | 10 < 10 is evaluated as False  10 < 100 is evaluated as True |
| >= | Used for checking if num1 is greater than or equal to num2 and is represented as num1 >= num2 | 10 >= 10 is evaluated as True  10 >= 100 is evaluated as False |
| <= | Used for checking if num1 is lesser than or equal to num2 and is represented as num1 <= num2 | 10 <= 10 is evaluated as True  100 <= 10 is evaluated as False |

**Assignment operators:**

| **Operator** | **Explanation** | **Example** |
| --- | --- | --- |
| = | Used for assigning a value to a variable | num=5  Here num is assigned the value 5 |
| += | Used as short hand assignment operator for addition | num=num+1 can also be represented as num+=1 |
| -= | Used as short hand assignment operator for subtraction | num=num-1 can also be represented as num-=1 |
| \*= | Used as short hand assignment operator for multiplication | num=num\*1 can also be represented as num\*=1 |
| /= | Used as short hand assignment operator for division | num=num/1 can also be represented as num/=1 |
| %= | Used as short hand assignment operator for modulo operation | num=num%1 can also be represented as num%=1 |

Next, you will discuss about the logical operators used in Python.

PAGE 6:

**Logical Operators:**

These operators are used to combine one or more relational expressions.

| **Operators** | **Description** |
| --- | --- |
| AND | Result will be true, if both the expressions are true. If any one or both the expressions are false, the result will be false |
| OR | Result will be true, even if one of the expression is true. If both the expressions are false, the result will be false |
| NOT | If the expression is true, result will be false and vice versa |

If A and B are two relational expressions, say A = (Num1>2000), B= (Num2>100), the result of combining A and B using logical operator is based on the result of A and B as shown below:

| **A** | **B** | **A and B** |
| --- | --- | --- |
| True | True | True |
| True | False | False |
| False | True | False |
| False | False | False |

| **A** | **B** | **A or B** |
| --- | --- | --- |
| True | True | True |
| True | False | True |
| False | True | True |
| False | False | False |

| **A** | **not A** |
| --- | --- |
| True | False |
| False | True |

Next, let us discuss the use of type-conversion in Python.

PAGE 7:

When you perform any operation on variables of different datatypes, the data of one variable will be converted to a higher datatype among the two variables and the operation is completed. This conversion is done by interpreter automatically and it is known as implicit type conversion. But Python does not support implicit type conversion and it will throw an error.

**Example:**

1. num1=10
2. num2="20"
3. result=num1+num2
4. print(result)

Output:

Traceback (most recent call last):  
  File "D:\Neon\Deepu\src\test.py", line 3, in <module>  
    result=num1+num2  
TypeError: unsupported operand type(s) for +: 'int' and 'str'

If you have to avoid this, then you have to explicitly convert the datatype of one variable into the required datatype to complete the operation. This is known as explicit type conversion.

**Example:**

1. num1=10
2. num2="20"
3. result=num1+int(num2)
4. print(result)

**Output:**

30

**Note:**

Programming languages define their own rules for implicit and explicit conversions. These rules do change from language to language.

Similarly, one has to be careful in explicit conversions as well. For example,

Converting a floating point value to integer would result in loss of decimal point values

A larger data type if converted to smaller data type will result in loss of data as the number will be truncated.

Next, let us see the use of comments in Python.

PAGE 8 :

Comments are the lines which are skipped during execution of a program.

There are two types of comments available in Python:

**Single line comment** which starts with ‘#’ symbol and extends till the end of line. Comments can start from the beginning of the line and middle of the line, but it should not be a part of string literal.

**Multi line comment** which starts with ''' or """ and ends with ''' or """ respectively. This type of comment is mainly used for documentation purpose.

**Example:**

1. '''
2. used for: Demonstrating comments
3. This is the first way of using multi-line comment
4. '''
5. """
6. used for: Demonstrating comments
7. second way of using multi-line comment
8. """
9. *#program to demonstrate explicit type conversion*
10. num1=10 *#variable of integer type*
11. num2="20" *#variable of string type*
12. result=num1+int(num2) *#using explicit conversion*
13. print(result)

**Output:**

30

Now you have to take up the quiz after which you will learn the various control structures used in Python.

==================================================

CONTROL STUCTURES IN PYTHON: (13 PAGES)