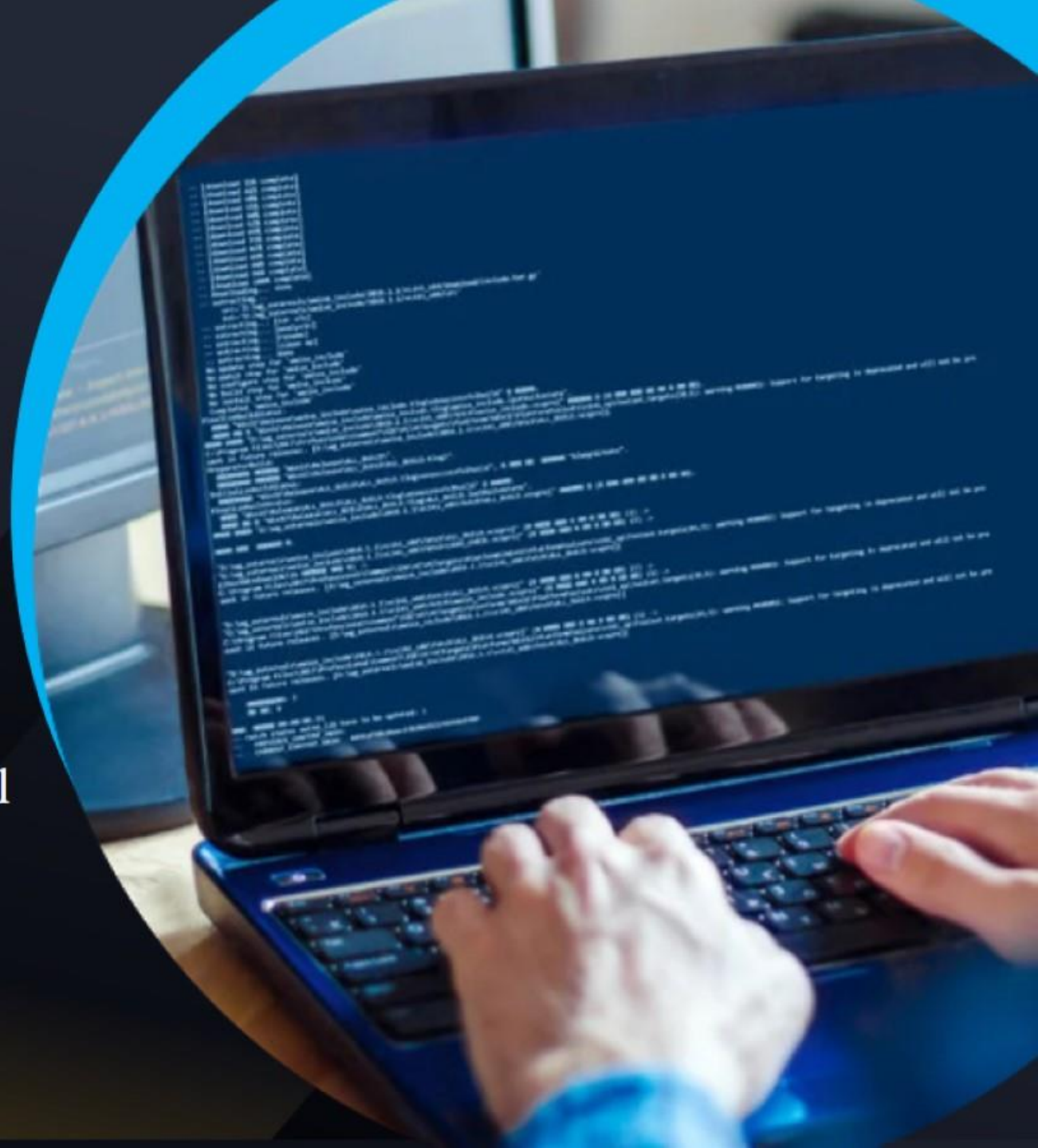


Programming Foundations in Python

Prepared by Dr. Hamada I. AbdulWakel

Lecture 2



Data Types & Input/Output Operations

Lecture 2

Prepared by
Dr. Hamada I. AbdulWakel

Lecture

02

RECAP: LECTURE 1

- Information processing and Problem-solving relationship.
- Algorithms and its features.
- Python programming language and its uses.
- Requirements for Pythons.

Lecture

02

LECTURE MAIN POINTS

- The software development life cycle (SDLC).
- The SDLC approaches/models.
- The input and output operations in Python.
- The data types and variables.
- The arithmetic expressions.

SOFTWARE DEVELOPMENT LIFE CYCLE

- The process of planning and organizing a program. It also known as Software Development Life Cycle (SDLC).



SOFTWARE DEVELOPMENT LIFE CYCLE

Cont'd

- **Planning:** outlines the scope of the problem and identify solutions. Resources, costs, time, and other aspects should be considered and determined here.
- **Analysis:** includes gathering all the specific details required for a new system as well as determining the first ideas for prototypes. Any ambiguities must be resolved in this phase.

SOFTWARE DEVELOPMENT LIFE CYCLE

Cont'd

- **Design:** Developers will first outline the details for the overall application, alongside specific aspects, such as its: User interfaces, Network requirements, and Databases.
- **Implementation:** the part where developers actually write code and build the application according to the earlier design documents and outlined specifications.

SOFTWARE DEVELOPMENT LIFE CYCLE

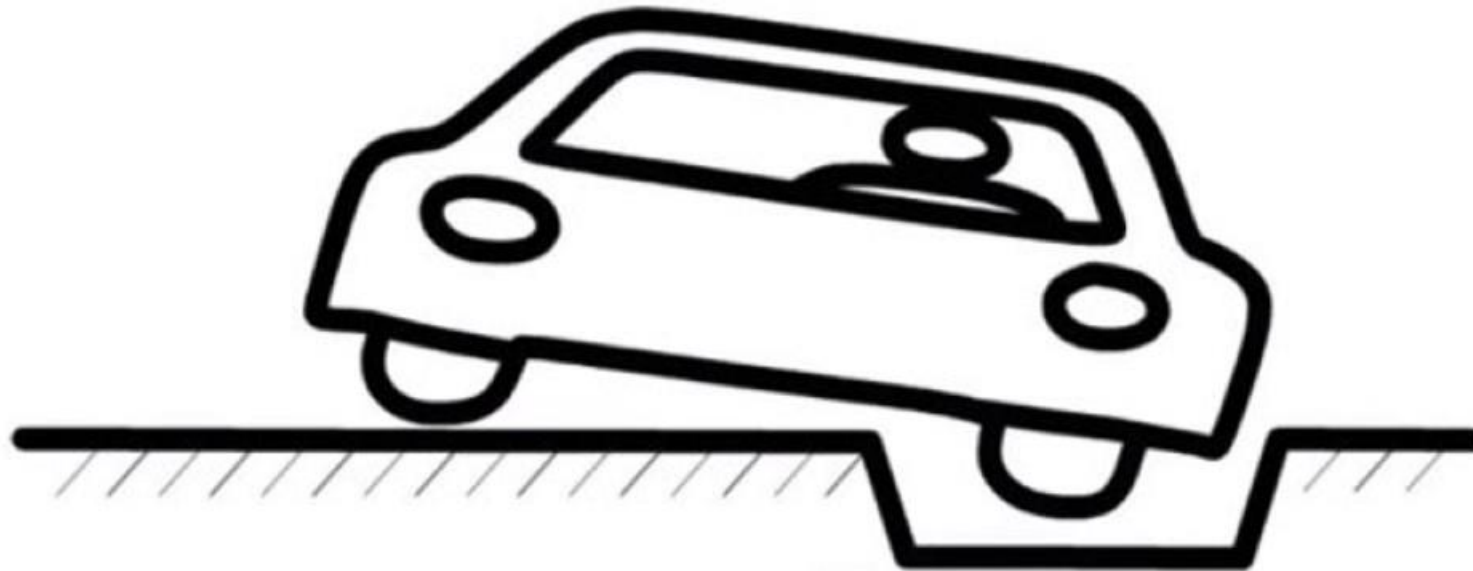
Cont'd

- **Testing:** all the pieces of code are tested to verify and validate a software product.
- **Maintenance:** once the system is deployed, any necessary upgrades, enhancements, and changes can be made, implementing new features into the operating software.

Lecture

02

SDLC APPROACHES



Pothole

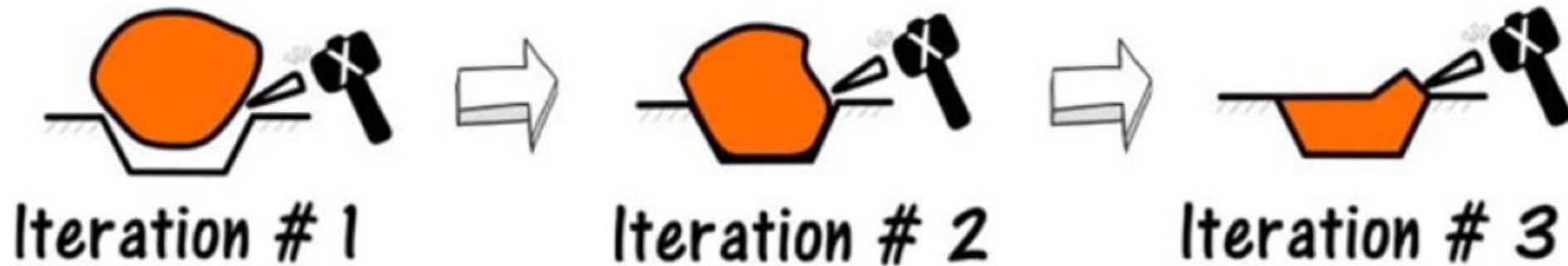
WATERFALL APPROACH



SDLC APPROACHES

Cont'd

ITERATIVE APPROACH



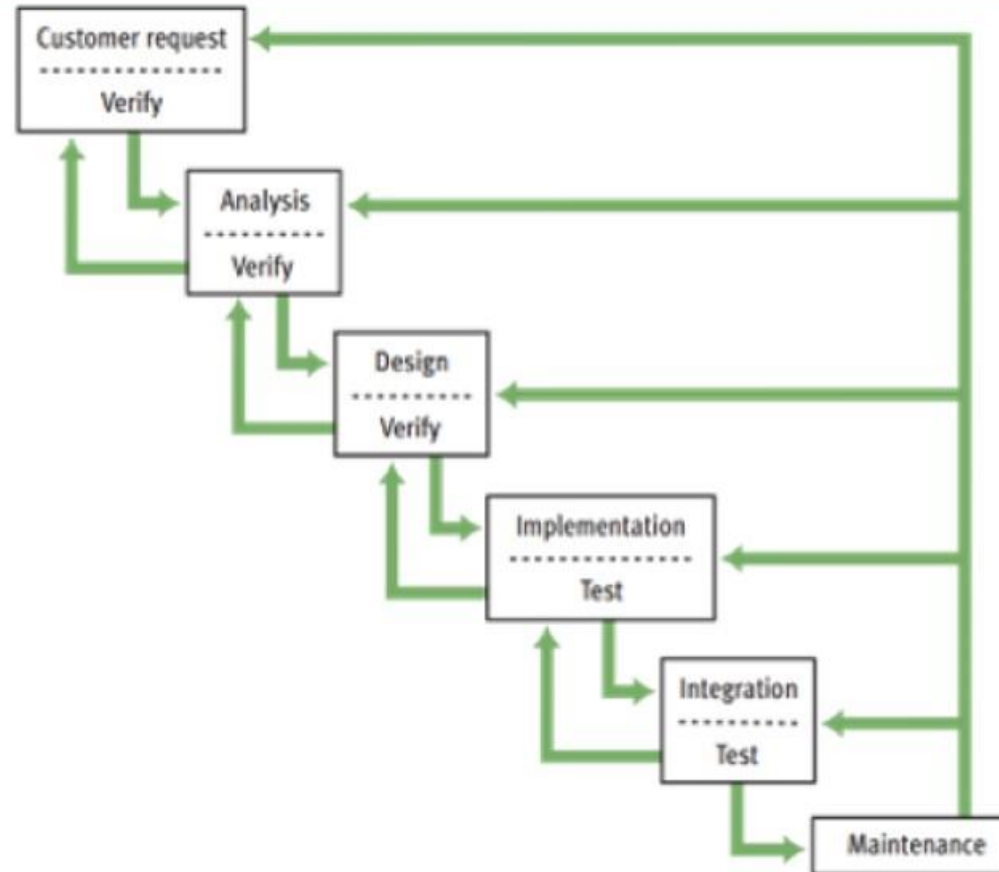
INCREMENTAL APPROACH



- **The Waterfall Model:**
 - One of the oldest models.
 - Recommending a more uniform/sequential software development.
 - Ensures that all required steps are completed before moving to the next stage. This helps project managers track the progress from start to finish.
 - It is difficult to adapt when flexibility is needed. Planning out change can be problematic.

SDLC APPROACHES

Cont'd

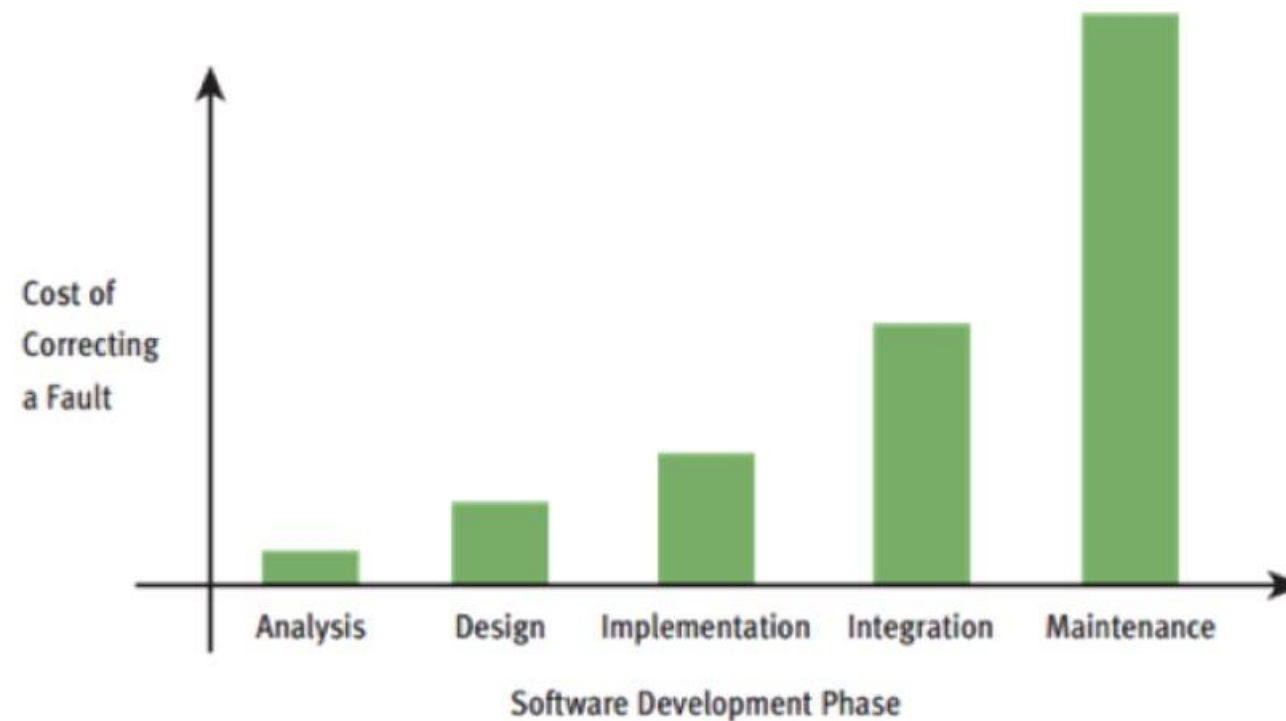


Lecture

02

SDLC APPROACHES

Cont'd



SDLC APPROACHES

Cont'd

- **The Iterative Model:**
 - It focuses on building solutions continuously by testing and modifying them.
 - It involves taking small steps (iterations) and making changes to a product quickly so that it becomes better and better over time.

- **The Incremental Model:**
 - The requirements are divided into multiple standalone modules of the SDLC.
 - Each module goes through the phases of SDLC.
 - Every subsequent release of the module adds function to the previous release.
 - The process continues until the complete system achieved.

Lecture

02

CASE STUDY: ANALYSIS INCOME TAX CALCULATOR

- The customer requests a program that computes a person's income tax.
 - Gross income required.
 - Flat tax rate 20%.
 - \$10000 standard deduction.
 - For each dependent, there is a \$2000 deduction.

Enter the gross income: 150000.00

Enter the number of dependent: 3

The income tax is **\$26800.00**

CASE STUDY: DESIGN INCOME TAX CALCULATOR

Cont'd

- An English-style algorithm is called **Pseudocode**.
 1. Input the gross income and number of dependents.
 2. Compute the taxable income using the formula
 3. $\text{Taxable income} = \text{gross income} - 10000 - (2000 * \text{number of dependent})$
 4. Compute the income tax using the formula
 5. $\text{Tax} = \text{taxable income} * 0.20$
 6. Print the tax

CASE STUDY: TESTING INCOME TAX CALCULATOR

Cont'd

- Error types:
 - **Syntax error:** occurred due to the mistakes while typing or do not follow the syntax of the specified programming language.
 - **Run-time error:** When the program is running, and it is not able to perform the operation.
 - **Logical error:** an error that leads to an undesired output.

CASE STUDY: TESTING INCOME TAX CALCULATOR

Cont'd

- **Test suite:** the logical grouping or collection of test cases to run a single job with different test scenarios.

NUMBER OF DEPENDENTS	GROSS INCOME	EXPECTED TAX
0	10000	0
1	10000	-600
2	10000	-1200
0	20000	2000
1	20000	1400
2	20000	800

DISPLAYING OUTPUTS IN PYTHON

- Almost all programs have solutions that needed to be displayed to the user through outputting data from the program.
 - Use the print() function.
`help(print)`
`print("Hello", "Ahmed")`
 - Change the default values of print() function.
`print("Hello", "Ahmed", sep="000")`
`print("Hello", "Ahmed", end="1111")`

Lecture

02

VARIABLES IN PYTHON

```
print("In 2020.")
```

```
print("Mohamed paid $120,000 for her house.")
```

```
print("Mohamed' house at 24 East Maple Street.")
```

VARIABLES IN PYTHON

Cont'd

```
print("In 2023.")
```

```
print("Aly paid $150,000 for her house.")
```

```
print("Aly' house at 12 Macca Street.")
```

VARIABLES IN PYTHON

Cont'd

```
Person_name = "Ahmed"
```

```
print("In 2023.")
```

```
print(Person_name, "paid $150,000 for her house.")
```

```
print(Person_name, "house at 12 Macca Street.")
```

VARIABLES IN PYTHON

Cont'd

```
Person_name = "Ahmed"  
Money = 150000
```

```
print("In 2023.")
```

```
print(Person_name, "paid", Money, "for her house.")
```

```
print(Person_name, "house at 12 Macca Street.")
```

VARIABLES IN PYTHON

Cont'd

```
Person_name = "Ahmed"  
Money = 150000
```

```
print("In 2023.")  
print(Person_name, "paid", Money, "for her house.")
```

```
Person_name = "Aly"
```

```
print(Person_name, "house at 12 Macca Street.")
```

DATA TYPES IN PYTHON

- Almost variable is a reserved memory location to store values.
- Python is a type-less language.

```
X = 10
```

```
X = "Hello world"
```

- The variable name refers to the memory location where its value store.
 - The id() function is used to get the memory location.

```
I = 10
```

```
print( id( I ) )
```

Lecture

02

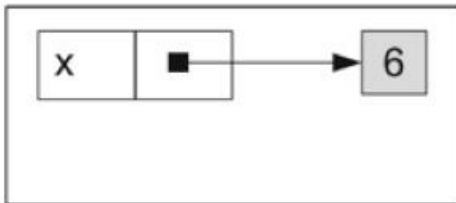
DATA TYPES IN PYTHON

Cont'd

memory

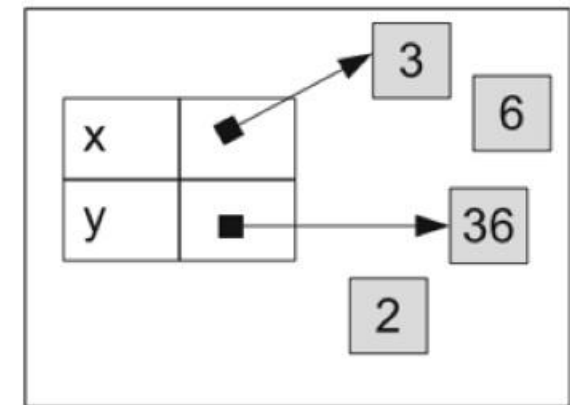
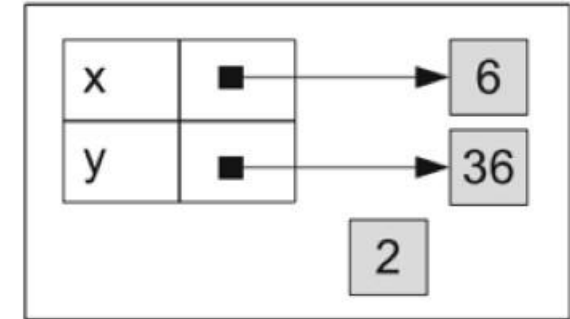


$x = 6$



$y = x ** 2$

$x = 3$



DATA TYPES IN PYTHON

Cont'd

- A variable name must begin with either a **letter** or an **underscore** (`_`).
- It contains any number of letters, digits, or other underscores.
- Variable names are **case sensitive**.

`X = 10``x = 10`

- Variable name could be unicode.

`س = 5`

DATA TYPES IN PYTHON

Cont'd

- The main data types:
 - **Strings**. A set of characters enclosed by double quotation " " or single quotation marks ' '
 - **Numbers**. A number written without quotations
 - **Booleans**. True or False value.
 - **List**.
 - **Tuple**.
 - **Dictionary**.

DATA TYPES IN PYTHON

Cont'd

- The `type()` function is used to know variable type:

```
X = 10
```

```
print( type( X ) )
```

- Escape sequences are characters used to express special characters as literals.

ESCAPE SEQUENCE	MEANING
<code>\b</code>	Backspace
<code>\n</code>	Newline
<code>\t</code>	Horizontal tab
<code>\\</code>	The <code>\</code> character
<code>\'</code>	Single quotation mark
<code>\"</code>	Double quotation mark

NUMERIC DATA TYPE

- Python's **int** data type consists of the integers from (-2^{31}) to $(2^{31} - 1)$.
- When the value of an integer exceeds these limits, Python automatically uses the **long** data type to represent it.
- Python uses **floating-point** numbers to represent real numbers.
- **Complex numbers** (e.g., $3 + 2j$) are also supported.

ARITHMETIC EXPRESSIONS IN PYTHON

- Expressions provide an easy way to perform operations on data values to produce other data values.
- An arithmetic expression consists of operands and operators.
 - Operands: the numbers or variables.
 - Operators: +, -, *, /

ARITHMETIC EXPRESSIONS IN PYTHON

Cont'd

Name	Meaning	Example	Result
+	Addition	15+20	35
-	Subtraction	24.5-3.5	21.0
*	Multiplication	15*4	60
/	Float Division	4/5	0.8
//	Integer Division	4//5	0
**	Exponentiation	4**2	16
%	Remainder	27%4	3

ARITHMETIC EXPRESSIONS IN PYTHON

Cont'd

Operator	Name	Example	Equivalent
<code>+=</code>	Addition assignment	<code>x += 4</code>	<code>x = x + 4</code>
<code>-=</code>	Subtraction assignment	<code>x -= 2</code>	<code>x = x - 2</code>
<code>*=</code>	Multiplication assignment	<code>x *= 5</code>	<code>x = x * 5</code>
<code>/*=</code>	Division assignment	<code>x /= 5</code>	<code>x = x / 5</code>
<code>//*=</code>	Integer division assignment	<code>x //= 5</code>	<code>x = x // 5</code>
<code>%*=</code>	Remainder assignment	<code>x %= 5</code>	<code>x = x % 5</code>
<code>**=</code>	Exponent assignment	<code>x **= 5</code>	<code>x = x ** 5</code>

ARITHMETIC EXPRESSIONS
IN PYTHON

Cont'd

- Rule of **priority**, that is some operator has highest priority than others.
 1. `()`
 2. `**`
 3. Unary operator
 4. `*`, `/`, `//`, `%`
 5. `+`, `-`
- Rule of **associativity**, evaluate expression of similar priority from left to right.

ARITHMETIC EXPRESSIONS IN PYTHON

Cont'd

- Example of arithmetic expressions.

```
print( 3.2 + 5 )
```

```
print( 4 / 2 )
```

```
print( 4 // 2 )
```

```
print( 7 % 2 )
```

```
x = 3
```

```
x += 4
```

```
print(x)
```

```
print( 2 + 5 / 3 * 7 )
```

```
print( ( 2 + 5 ) / 3 * 7 ) #note the difference in the result
```

FUNCTIONS WITH NUMERIC DATA TYPES

- Numeric functions:

```
print( bin(50) )  
print( oct(50) )  
print( hex(50) )  
print( int(0x32) )  
print( abs(-5) )  
print( pow(2, 3) )  
print( max(4, 5) )  
print( min(4, 5) )  
print( round(3.7) )
```

COMMENTS IN PYTHON

- **A comment:** a piece of program text that the interpreter **ignores**. It provides useful **documentation** to programmers.

```
# this is a single line comment
```

```
""" this is  
    A multi-line comment """
```

GETTING INPUTS IN PYTHON

- Almost all programs **interact with the user** through giving data to the program to process it and display other data.
 - Use the `input()` function.
`input( )`
`input( "Enter your name: " )`
`name = input( "Enter your name: " )`
 - Change the type of the value given from the user.
`stdCount = int( input( "Enter the number of students: " ) )`
`stdAge = float( input( "Enter student age: " ) )`

Lecture

02

SIMPLE CALCULATOR PROGRAM

```
num1 = input( "Enter the first number: " )
```

```
num2 = input( "Enter the second number: " )
```

```
result = num1 + num2
```

```
print( result )
```

SIMPLE CALCULATOR PROGRAM

Cont'd

```
num1 = int( input( "Enter the first number: " ) )  
  
num2 = int( input( "Enter the second number: " ) )  
  
result = num1 + num2  
  
print( result )
```

Lecture

02

QUESTIONS FOR ONLINE DISCUSSION

- Write the program of the income tax calculator discussed in the lecture.
- Write a program that asks the user to enter two integers. Have the program output the two integers and the result when the first number entered is raised to the power of the second number entered.
- Complete the following code and expect the output.

```
a = 5
```

```
b = 10
```

```
sum = a ... b
```

```
print(sum)
```



Thank You

