Integrated Learning Programme (ILP) 2024 : Computing

Alakesh Kalita

Week 1 Session 1(W1S1)
13 May 2024

A quick word about me

Dr. Alakesh Kalita

Current Affiliation: Lecture, Singapore University of Technology and Design, Singapore

Ph. D. : Indian Institute of Technology Guwahati, India

Research domain: Computer Networks, Internet of Things

Previous work experience: National University of Singapore, Singapore

Profile link: https://istd.sutd.edu.sg/people/faculty/alakesh-kalita/

https://alakesh-kalita.github.io/

Email IDs: alakesh_kalita@sutd.edu.sg; alakesh.kalita1025@gmail.com

Course Objectives

1. We will learn Python Programming Language
    a. Sometimes, we will also learn the syntax of C, C ++, and Java.

2. Building Logic to solve computational problems

    a. Computation without programming language
        i. Make the number of negative 1 —> -1.
        ii. Find the biggest number among the given three numbers: 13, 45, -135
        iii. Swap two numbers X = 65, Y = 18 —> X = 18, Y= 65, using/without using another variable “Z”

3. Use of proper data structure to solve different computation problems

4. How to design good algorithms.

Course Impact

1. You will have idea about programming languages

2. You will know how to solve a given problem (Logically)

   - Can be used with any programming language
   - You need to remember the syntax of programming languages
     - `print("Hello World")` in Python
     - `printf("Hello World");` in C
     - `cout << "Hello World";` in C++
     - `System.out.println("Hello World");` in Java

Building logic is much more important. Syntax you can google it :)

3. Nowadays, all the major courses such as any Engineering field, Science (PCM), Economics, etc., use computer programming for……. (we discuss later :))

Topics for Week 1

W1S2:

- What is Programming Language, Logic, Algorithm?
- Configuring and installing of Python
- Introduction to Jupyter notebook, Google Colab, etc.
- Variables, math operators, comments, operator precedence, overloading
- Input from users
- Type casting

W1S2:

- If, elif, else, while, break statements
- For loops, generators and recursion
- Various in-built functions

Topics for Week 2

W2S1:

- What is data and data structure?
- List data structure
- Loops with list
- Function
- Recursion

W2S2:

- String
- Tuple
- Sets
- Dictionary
- Algorithm Complexity
- Algorithm Optimization

Topics for Week 3

W3S1:

- OOP concept
- Class and Object
- Attribute, method, special methods etc.
- Inheritance
- Polymorphism
- Encapsulation

W3S2:

- Error Handling
- Plotting
- Numpy library

Topics for Week 4

W4S1:

- Memory management and lists: aliasing, shallow and deep copies
- The Numpy library (part 1): arrays, math functions, etc.
- About the import procedure
- Project organizing
- Mini-project

W4S2:

- Recap
- Mini-project

Delivery

- 2 x 4.5hrs - lessons per week
- Lessons include mostly practical with a bit of theory (PPT slides)
- Practice activities (in Jupyter Notebooks)

Teaching materials

- PPT/PDF contain the lecture materials (PPT preferred)
- Activities notebooks and their answers
- eDimension
- The teaching materials will be uploaded on eDimension and made available on the same day.

Homeworks, extras and exams

- No Homeworks :); No Grading
- But, we will have lots of class activities :D
- Extra challenges: advanced activities will be discussed in class. (You can try at home. Solutions will not be provided :(but can be discussed :))

Let’s get started…

What is Programming?

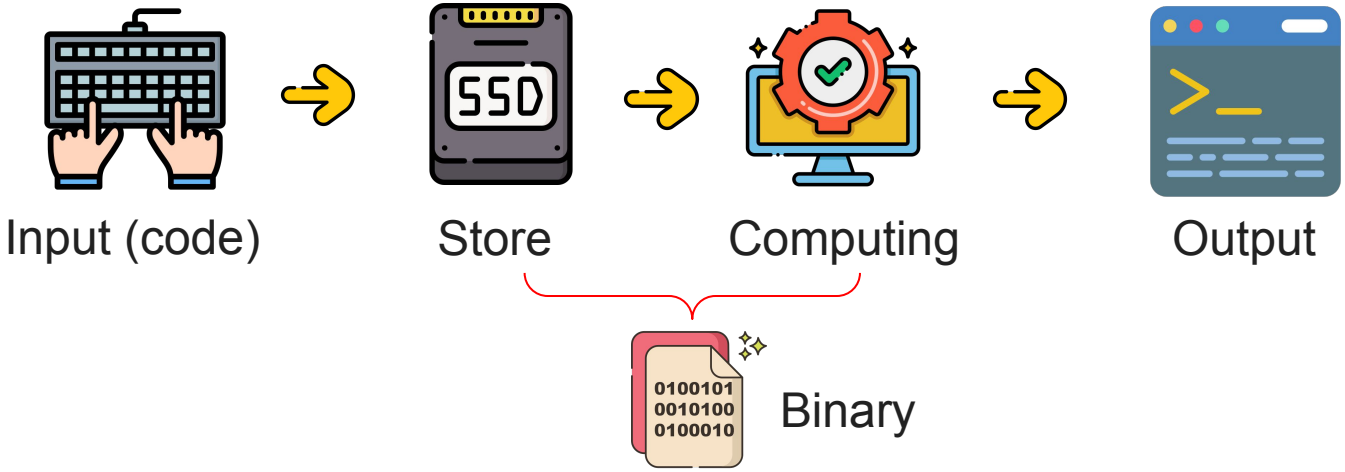
Definition: Programming is the process of instructing computing devices to perform tasks or operations

- by providing it with a set of instructions, known as code.

What's your name?

Input (Listen) → Remember → Process (neurons) → Output (speak)

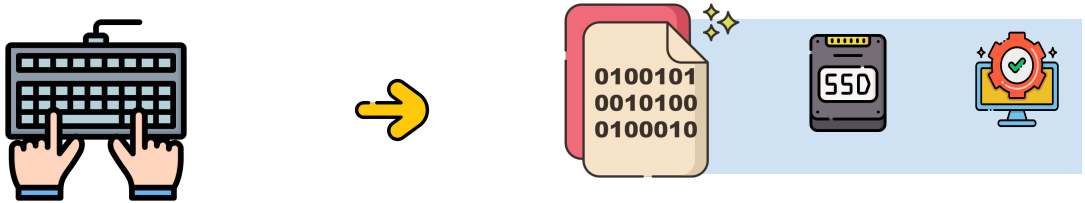
What is Programming?



Our computing devices (computer) understand (store and process) only binary numbers *i.e.*, 0 and 1

- "Alakesh" is in binary form 01000001 01101100 01100001 01101011 01100101 01110011 01101000

Contd…



Input → Binary

Should be done in such a way that a computer can understand easily → We need programming language

✗ Native language

Contd…

- So, we can say that programming languages provide a common syntax for everyone who uses them.

- This gives birth to different programming languages, such as
  - C
  - C++
  - Java
  - Python
  - HTML
  - and many more ……. but

Native language

Applications of Programming

Software Development: For developing software applications such as web apps, mobile apps, desktop software, and games.

Web Development: Programming languages like HTML, CSS, JavaScript, and Python are used in web development to create dynamic and interactive websites.

Artificial Intelligence and Machine Learning: Programming plays a crucial role in building AI and machine learning models for tasks such as image recognition, natural language processing, and predictive analytics.

Many more:

Program can do

Automation: Programming allows tasks to be automated, reducing manual effort and increasing efficiency.

- Car parking gates in Singapore
- Changi Airport

Problem Solving: It equips individuals with the ability to analyze problems and develop solutions systematically.

- Email Service, Bluetooth

Innovation: Programming drives innovation by enabling the creation of new software, technologies, and digital experiences.

- PayLah !!!

What is logic?

It is the foundation of problem-solving and critical thinking.

- How you are going to solve a problem?
  - Methods/Approach
- What are the constrained?
  - Ex: In adding two numbers problem, input should not be a string
- What would be the output?
  - What's the user want?
- How to make it efficient?
  - Less lines of code/instruction :)
  - Time and Memory :D

Logic Exercises

1) Make a given number negative. Ex. 1→ -1.
2) Find the biggest number among the given three numbers: 13, 45, -135
3) Determine if a given number is even or odd.
4) Find 3 + 6 + 9 + 12 + …. + 60 = ?
5) Finding the Maximum number among these numbers : 10, 3, 2 ,1, 77, 108
6) Check if a given year is a leap year or not.
7) Calculate the sum of the digits of a given integer.
8) Swap two numbers X = 65, Y = 18 —> X = 18, Y= 65, using another variable “Z”
9) Swap two numbers X = 65, Y = 18 —> X = 18, Y= 65, without using another variable.

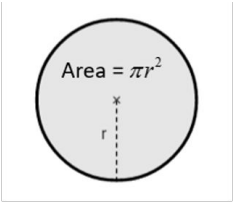
Introduction to Algorithm

- Before solving a problem, we write down a possible solution on piece of paper in our native language.
- We call it Algorithm.

Algorithm	Program
For human being	For Computer
Native Language	Programming language, syntax and rules must be followed
Piece of Paper and Pen	Programming Environment / Hardware
Independent of any specific programming language	Dependent on Programming environment

Algorithm: definition

Definition (Algorithm): An algorithm is a finite sequence of instructions, typically used to solve a class of problems or perform a computation task.



Example: Compute the area of a circle with radius 10 using a calculator.

Python Code

```
import math

# Define the radius of the circle
radius = 10

# Calculate the area of the circle
area = math.pi * radius**2

# Print the result
print("The area of the circle with radius", radius, "is:", area)
```

Algorithm

- Type **10**,
- Type multiply key,
- Type **10**,
- Press equal/enter key,
- Type multiply key,
- Type π,
- Press equal key again,
- Display output

The life of a programmer

1. Identify a **problem**.
2. Come up with an **algorithm** to solve it.
3. Represent the algorithm as a **program**, using a chosen programming language.
4. **Execute** the program on a computer!
5. **Find Error** : Goto step 2
6. No error (receive desired output)
7. Deliver the **output**
8. Goto Step 1 :)

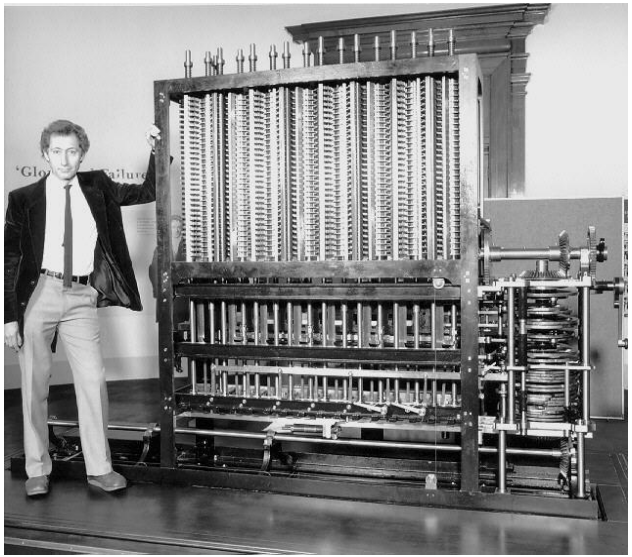
source: Google Image

Inventor of Computer

Who is considered the inventor of computers and “programming”?

Charles Babbage (1791–1871):

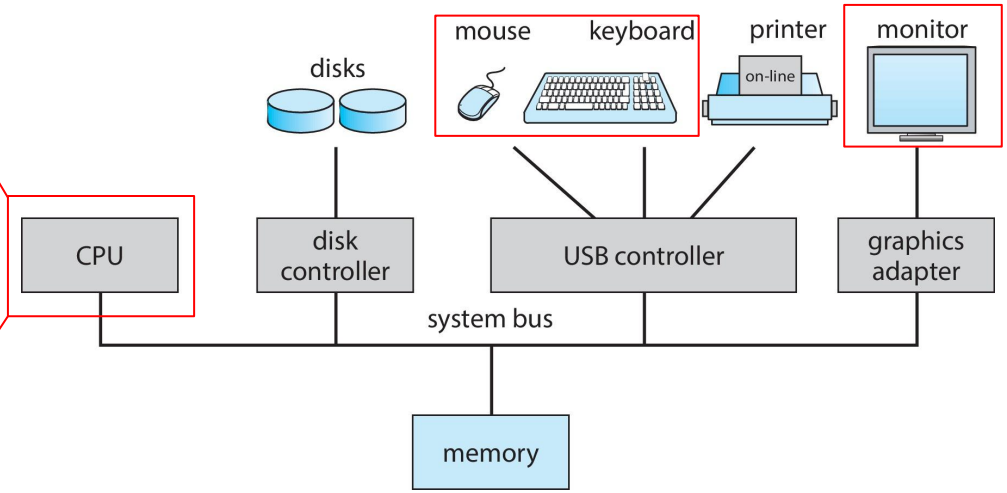
- Often referred to as the "*father of the computer*,"
- English mathematician, philosopher, inventor, and mechanical engineer.
- Babbage designed the first mechanical computer known as the "Analytical Engine" in the 1830s.



source: Google Image

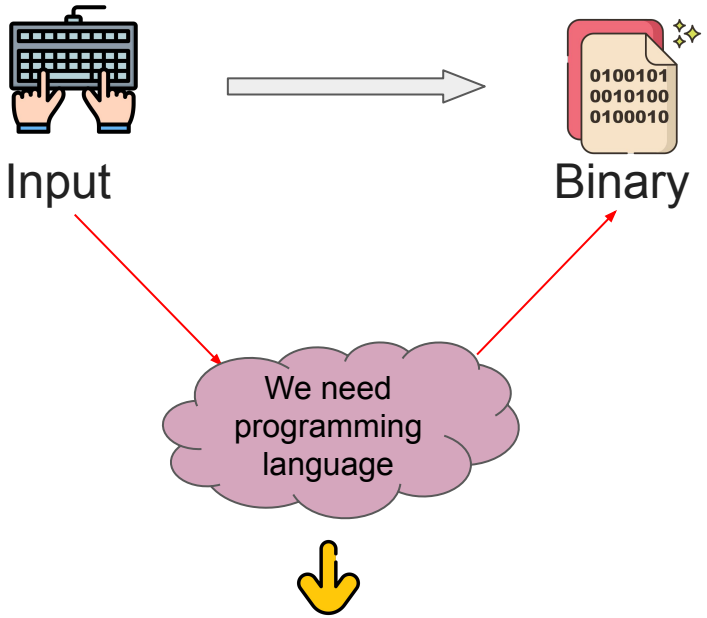
A typical computer system

source: YouTube



source: Abraham's OS Book

Contd…



- High-level language
- Low-level language

High-level language	Low-level language
Closer to human language	Closer to machine code
Easier to read, write, and maintain	Require a deeper understanding of computer architecture
Offer built-in data structures and functions	Offer more control over hardware resources, (memory, registers)
Slower	Faster
Ex: Python, Java, C++, JavaScript, Ruby.	Ex: Assembly language, Machine code.

Note: High-level languages converted to assembly code by **Compiler** before final execution.

python™: what is it?

High-Level, Interpreted Language:

- Its interpreted nature allows for rapid development and easy debugging.

Simple and Readable Syntax:

- Its clean syntax and use of indentation make code easy to understand and maintain.

Dynamic and Strong Typing:

- Python is dynamically typed, eliminating the need for explicit variable declarations such as int, string, float.

Rich Standard Library and Ecosystem:

Created by **Guido van Rossum** and first released in 1991.

Why learn Python?

- **Python is widely used in IT companies.**
- Also, **high interoperability with other languages:** Jumping to or including another language (Java, C, SQL, etc.) is easy, once you know Python.
- So many libraries available

Overall, Python is a good entry point for beginners in both computer science and other domains, and therefore, the first language we teach in SUTD.

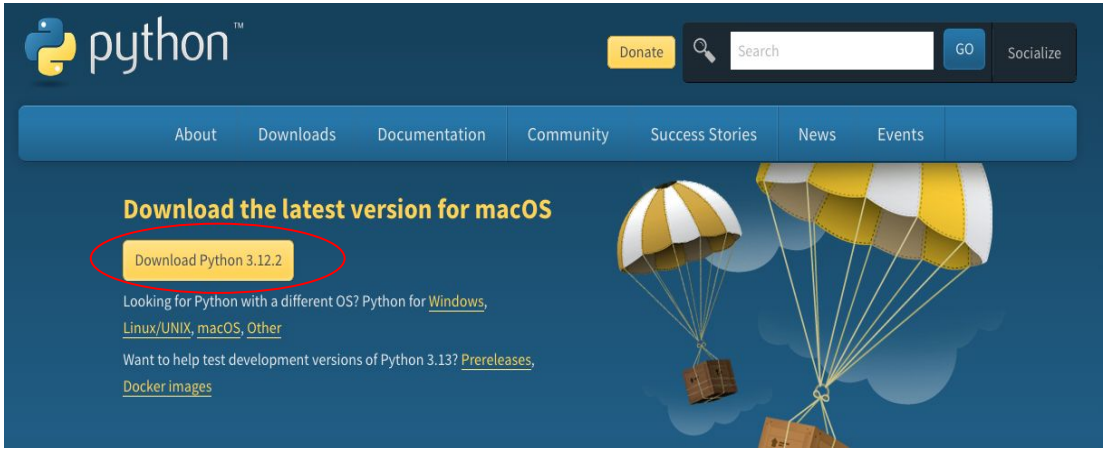
Any Question?

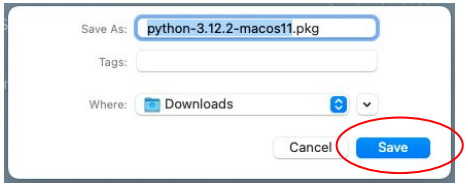
Next→ We are going to install Python

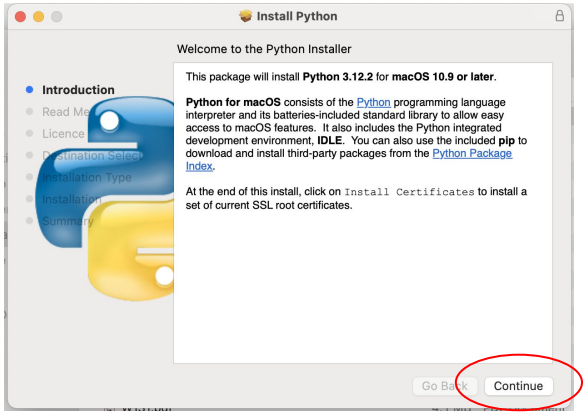
Installing Python

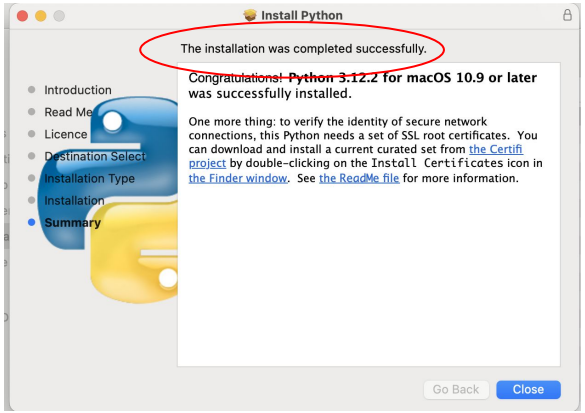
- In this class, we will use the latest version of Python.
- You can download Python on your machine using the below link:

  https://www.python.org/downloads/

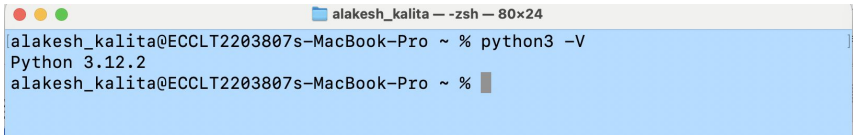








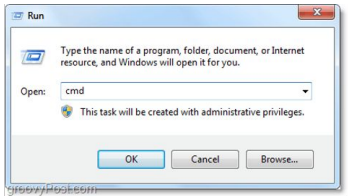
- To verify your installation
  - Open Terminal
  - Type python3 -V



MacOS

Terminal

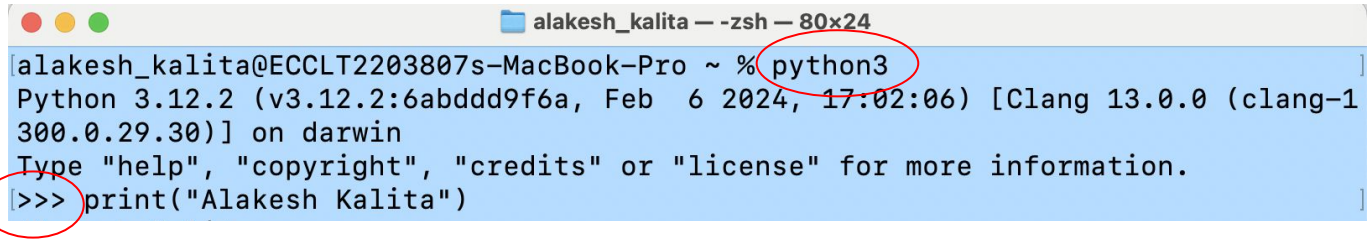
Windows OS



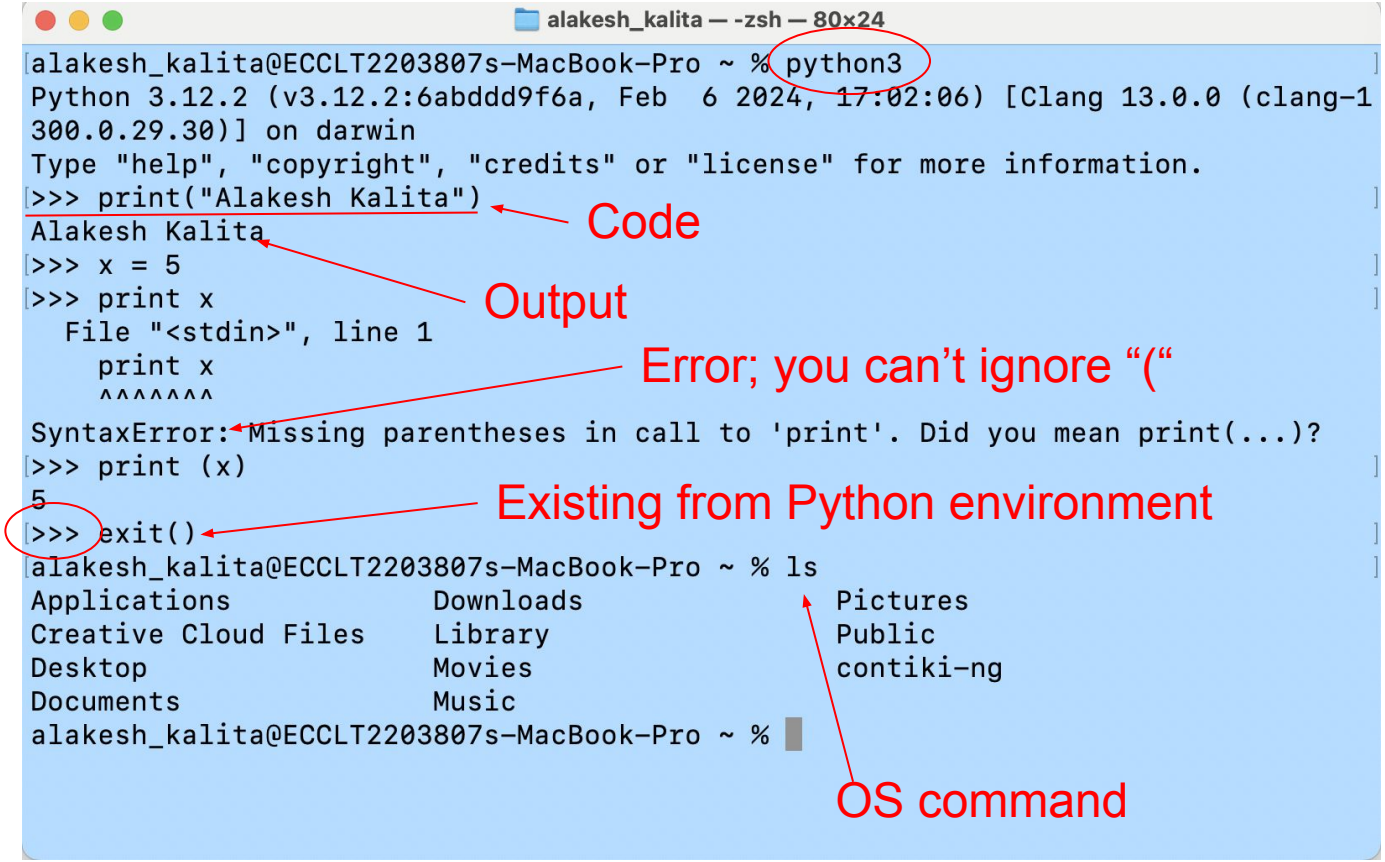
OS Environment: If you type any command after launching the **terminal**, that command would be handle by **Operating System** (OS)

```
alakesh_kalita@ECCLT2203807s-MacBook-Pro ~ % ls
Applications            Downloads               Pictures
Creative Cloud Files    Library                 Public
Desktop                 Movies                  contiki-ng
Documents               Music
alakesh_kalita@ECCLT2203807s-MacBook-Pro ~ %
```

Python environment: To get into Python environment, you need to execute the below command: `python3`



Line starts with three arrows (= Python mode)


alakesh_kalita — -zsh — 80×24
alakesh_kalita@ECCLT2203807s-MacBook-Pro ~ % python3
Python 3.12.2 (v3.12.2:6abddd9f6a, Feb 6 2024, 17:02:06) [Clang 13.0.0 (clang-1
300.0.29.30)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Alakesh Kalita")
Alakesh Kalita
>>> x = 5
>>> print x
File "<stdin>", line 1
print x
^^^^^^^
SyntaxError: Missing parentheses in call to 'print'. Did you mean print(...)?
>>> print (x)
5
>>> exit()
alakesh_kalita@ECCLT2203807s-MacBook-Pro ~ % ls
Applications Downloads Pictures
Creative Cloud Files Library Public
Desktop Movies contiki-ng
Documents Music
alakesh_kalita@ECCLT2203807s-MacBook-Pro ~ %
Code
Output
Error; you can't ignore "("
Existing from Python environment
OS command

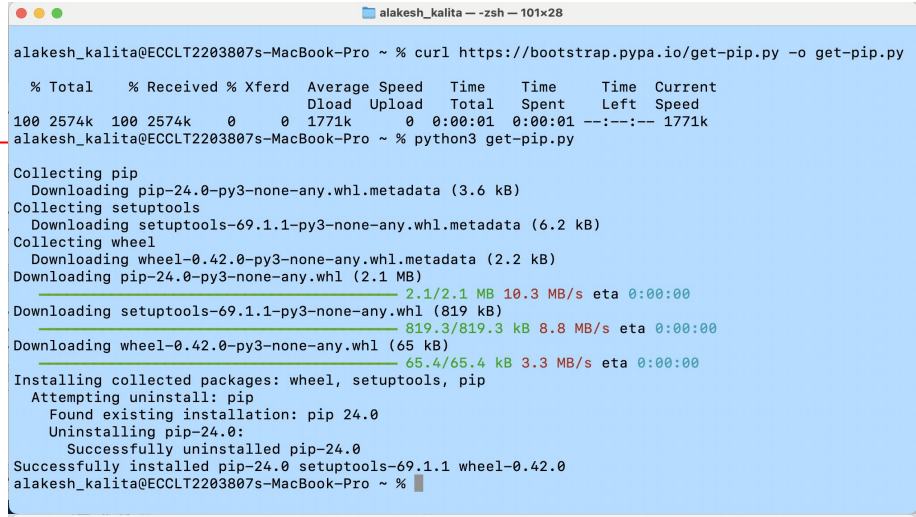
What is GUI ?

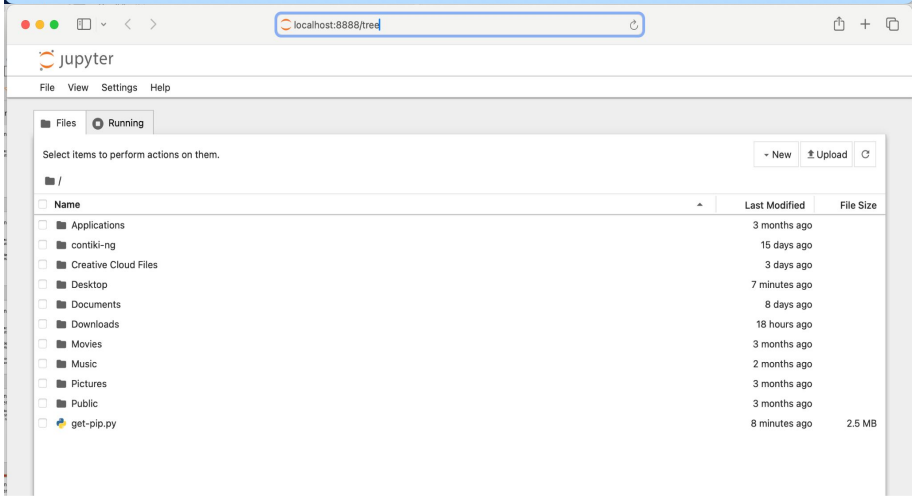
- Writing program on “Console” is user friendly/convenient, specially, when you write long and complex program

- Therefore, we prefer “Graphical User Interface” (GUI) based Python (any) programming language environment
  - Interactive Development Environment (IDE), which makes the coding easier for us.
  - In this course, I suggest to use Jupyter Notebook and Google Colab, but you might look online for other IDEs if you want!

Installing Jupyter Notebook

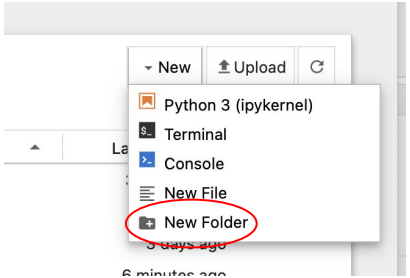
- Download and install Pip on macOS
  - `curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py`
  - `python3 get-pip.pyclear`
- Download Jupyter Notebook
  - `pip install notebook`
- Run Jupyter Notebook
  - `jupyter notebook`

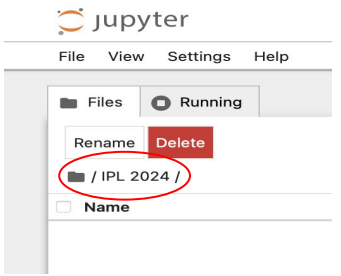


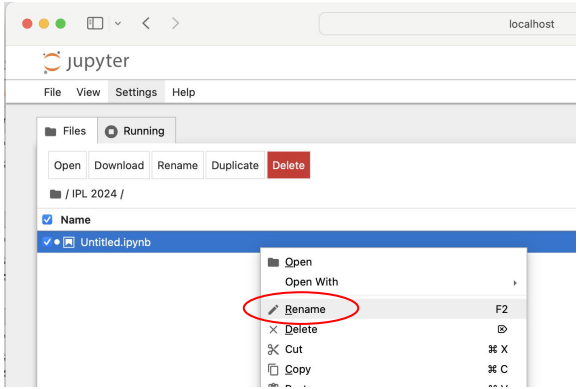


Working on Jupyter notebook

- Create new folder/directory
  - ILP 2024
- Go inside the folder
- Goto New→Python 3 (ipykernal)
- Rename the file “W1S1.ipynb”

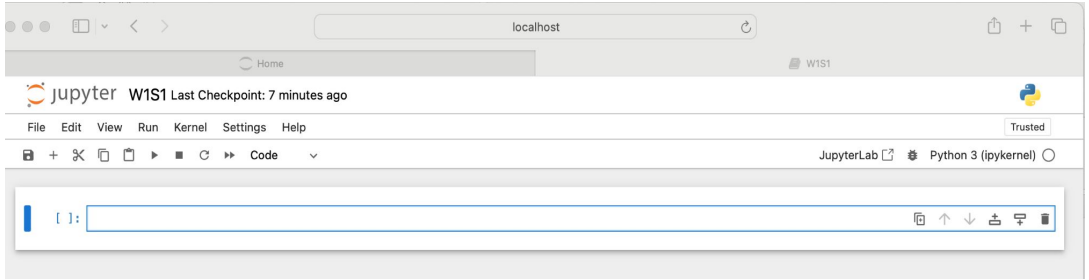




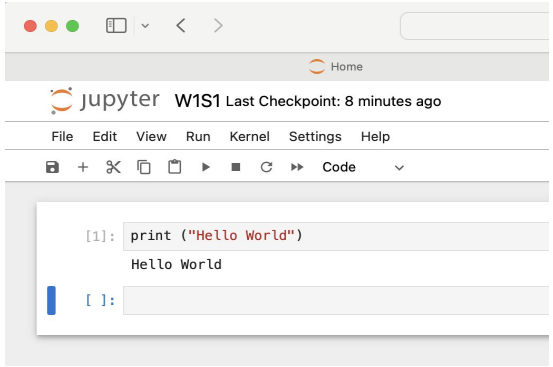


Hello World ! My first program

- Double click on W1S1.ipynb



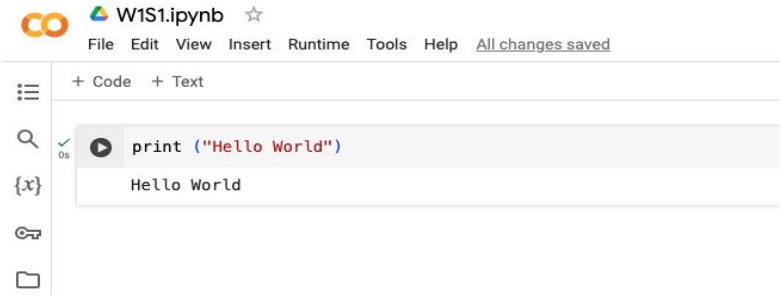
- Run your first program
  - SHIFT + Return
  - `print ("Hello World")`



Congratulations !!!

Google colab

- Sign Up for a google account and Log in
- Search “Google Colab” on Google
  - Or Goto https://colab.google/
- Click on “New Notebook”
- Rename the file and start writing your code



Jupyter Notebook/ Console	Google Colab
Run on your local machine	Run on Google’s server
No need of Internet connection	Need Internet connections
Can use your system resources	Limitations on CPU, RAM, DISK, and GPU
Faster	Depend on your Internet speed
Need to install Python, Notebook, Packages	Ready to use

Congratulations, you now have a Python-compatible machine, ready to run!

Feel free to play around a bit more if you want!

How does Python work?

- `print ("Hello World")`
  - `print()` function in Python is a built-in function used to display output
  - Takes zero or more **arguments**
  - `print()` function is called with the argument "**Hello World**".
  - `print()` function converts the arguments to strings if they are not already strings.
  - `print()` function calls `sys.stdout.write()` method for writing to the standard output stream.

- Everything is done by Python Interpreter (Compiler)

`sys.stdout.write()`

```
section .data
    message db 'Hello, World!', 10    ; Message
to be written, followed by newline character
(10)

section .text
    global _start

_start:
    ; Prepare the arguments for the write
system call
    mov eax, 4                        ; syscall
number for sys_write
    mov ebx, 1                        ; file
descriptor 1 (stdout)
    mov ecx, message                  ; address of
the message to be written
    mov edx, 13                       ; number of
bytes to write (length of message)
    int 0x80                          ; invoke
syscall

    ; Exit the program
    mov eax, 1                        ; syscall
number for sys_exit
    xor ebx, ebx                      ; exit code 0
    int 0x80                          ; invoke
syscall
```

Thanks to creator and developer of Built-in functions (Libraries) !!!

What is String?

- In Python, a string is a sequence of characters enclosed within either single quotes (' ') or double quotes (" ").
  - `print ("Hello World")`
- Guess the outputs →

```
print ("Hello World")
```

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

```
print("5+5")
```

```
print("Hello World")
Hello World
print(5+5)
10
print("5+5")
5+5
```

- You can add (concat) two strings together, but can't add a string with other non-string

```
print("Hello World: " + "5")
Hello World: 5
print("Hello World: " + 5)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[29], line 1
----> 1 print("Hello World: " + 5)

TypeError: can only concatenate str (not "int") to str
```

Thanks to interpreter

Literal in Python

In Python, `literals` are data that are specified directly in the code.

- They are constant values of some built-in types like numbers, strings, and booleans.
- For example, 5, "hello", and True are all literals.

- `print("Hello World")`
  - "**Hello World**" is a sequence of characters enclosed in double quotes and is treated as a **string literal.**
  - The `print()` function outputs the string "**Hello World**" to the console.

- `print(5+5)`:
  - Python first evaluates the expression 5+5.
  - In this expression, 5 and 5 are **integer literals**, and **+** is the **addition operator**.
  - The `print()` function then takes the result (**10**) as its argument and outputs the **string "10"** to the console

Variables, Data types, and Operators

Variables: Variables are created by assigning a value to a name.

- Must be defined before using it

```
x = 5
```

```
print(x)
```

```
5
```

```
print(y)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
Cell In[32], line 1
----> 1 print(y)

NameError: name 'y' is not defined
```

Thanks to interpreter

Literals	Variables
Constant	Mutable; changeable
Values are fixed	Values can change
Directly specified in the code	Assigned a name in the program

Variables, Data types, and Operators

Data types: Python supports several built-in data types, including integers, floats, strings, booleans, lists, tuples, and dictionaries.

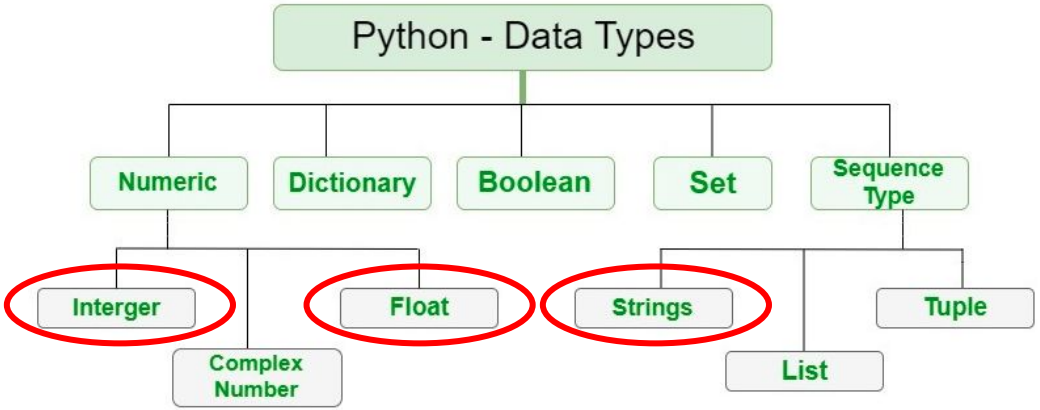
- `type(variable_name)`, will give you the data type

```
x = 99
y = 0.5
string1 = "Hello World"
print(type(x))
print(type(y))
print(type(string1))
```

```
<class 'int'>
<class 'float'>
<class 'str'>
```

Note: In C, C++, Java, we need to mention the type of variable during its declaration; eg: `int x`, `float y = 0.5`

Data types in Python



Arithmetic Operators

There are seven arithmetic operators in Python as follows:

Operator	Description	Syntax
+	Addition: adds two operands	x + y
–	Subtraction: subtracts two operands	x – y
*	Multiplication: multiplies two operands	x * y
/	Division (float): divides the first operand by the second	x / y
//	Division (floor): divides the first operand by the second	x // y
%	Modulus: returns the remainder when the first operand is divided by the second	x % y
**	Power (Exponent): Returns first raised to power second	x ** y

```
x = 5
y = 2.0
```

```
print (x+y)
```
```
7.0
```

```
print (x-y)
```
```
3.0
```

```
print (x*y)
```
```
10.0
```

```
print(x/y)
```
```
2.5
```

```
print(x//y)
```
```
2.0
```

```
print(x%y)
```
```
1.0
```

```
print(x**y)
```
```
25.0
```

Comparison Operators

- **Equal to (==)**:
  - Returns True if the operands are equal, otherwise False. Example: x == y
- **Not equal to (!=)**:
  - Returns True if the operands are not equal, otherwise False. Example: x != y
- **Greater than (>)**:
  - Returns True if the left operand is greater than the right operand, otherwise False. Example: x > y
- **Less than (<)**:
- **Greater than or equal** to (>=):
- **Less than or equal to** (<=):

Note: Line starts with "**#**" is not considered as instruction (so not executed), called **comment.**

```
x = 10
y = 5

# Equal to
print(x == y)  # Output: False

# Not equal to
print(x != y)  # Output: True

# Greater than
print(x > y)   # Output: True

# Less than
print(x < y)   # Output: False

# Greater than or equal to
print(x >= y)  # Output: True

# Less than or equal to
print(x <= y)  # Output: False
```

```
False
True
True
False
True
False
```

Logical Operator

- Logical operators are used to perform **logical operations** on **boolean values**.

- Python supports three main logical operators: `and`, `or`, and `not`.

A	B	A and B	A or B
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

```python
# Logical Operator
# Define variables
A = True
B = False

# Logical AND operation
result_and = A and B
print(result_and)  # Output: False

# Logical OR operation
result_or = A or B
print(result_or)    # Output: True

# Logical NOT operation
result_not = not A
print(result_not)
```

```
False
True
False
```

Exercises

Do it now

1. If all cats can fly, and Tom is a cat, can Tom fly?

2. If it is raining outside and I don't have an umbrella, will I get wet if I go outside?

3. If all apples are fruits and some fruits are red, can we conclude that some apples are red?

4. If every student in the class passed the exam and Jane is a student, did Jane pass the exam?

5. If it is not Monday and I don't have to go to work, can I sleep in late?

```python
# Solution to Question 1
all_cats_can_fly = True
tom_is_a_cat = True
can_tom_fly = all_cats_can_fly and tom_is_a_cat
print("Can Tom fly?", can_tom_fly)
```

```
Can Tom fly? True
```

```python
# Solution to Question 2
is_raining_outside = True
have_umbrella = False
will_get_wet = is_raining_outside and not have_umbrella
print("Will I get wet?", will_get_wet)

# Solution to Question 3
all_apples_are_fruits = True
some_fruits_are_red = True
some_apples_are_red = all_apples_are_fruits and some_fruits_are_red
print("Are some apples red?", some_apples_are_red)

# Solution to Question 4
all_students_passed_exam = True
jane_is_a_student = True
did_jane_pass_exam = all_students_passed_exam and jane_is_a_student
print("Did Jane pass the exam?", did_jane_pass_exam)

# Solution to Question 5
not_monday = False
dont_have_to_go_to_work = True
can_sleep_in_late = not_monday and dont_have_to_go_to_work
print("Can I sleep in late?", can_sleep_in_late)
```

```
Can Tom fly? True
Will I get wet? True
Are some apples red? True
Did Jane pass the exam? True
Can I sleep in late? False
```

Operator precedence

- By default, operations are computed from **left to right**
- However, operation precedence applies: **exponentiations > multiplications/divisions > additions/subtractions.**
- As in mathematics, **parentheses** can be used to "**force**" precedence.

```python
#Operator predence examples

a = 3
b = 2
c = 4

# Arithmetic operations are evaluated first, then assignment
result_arithmetic = a + b - c   # Left to Right
print(result_arithmetic)

result_arithmetic = a + b * c  # Multiplication (*) has higher precedence than addition (+)
print(result_arithmetic)

result_arithmetic = (a + b) * c  # parentheses can be used to "force" precedence.
print(result_arithmetic)

result_arithmetic = a*b**c
print(result_arithmetic)

# Comparison operators
result_comparison = a + b < c * 2  # Arithmetic operations are evaluated first, then the comparison
print("Result of comparison operation:", result_comparison)  # Output: True

# Logical operators
result_logical = (a > b) and (c < b or a == 3)  # Comparison operations are evaluated first, then logical operations
print("Result of logical operation:", result_logical)  # Output: True
```

```
1
11
20
48
Result of comparison operation: True
Result of logical operation: True
```

Operator Overloading

The **behavior of a basic math** operator (+, -, *, etc.) might vary, depending on the **types of the variables** involved in the calculation.

This is called **operator overloading**, and is a perfectly important feature of Python programming.

```python
# Operator overloading for addition with integers
num1 = 5
num2 = 10
result_int = num1 + num2
print(result_int)  # Output: Result with integers: 15

# Operator overloading for addition with strings
str1 = "Hello "
str2 = "world"
result_str = str1 + str2
print(result_str)  # Output: Result with strings: Hello world
```

```
15
Hello world
```

Thanks to Python

Quick question :)

What is the difference between “=” and “==” operators?

Programming Test - 1

Implement Now :)

1) Make a given number negative. Ex. 1→ -1.
2) Determine if a given number is even or odd.
3) Find 3 + 6 + 9 + 12 + …. + 60 = ?
4) Compute the area of a circle with radius 10. Hint: `my_pi = 3.14`

Maybe your code is wrong :D

Revisiting Variables

- **Variables:** It is created by assigning a value to a name.

```
x = 10
```

- Variables names can include any combination of letters (both lowercase and uppercase), digits and underscore symbols (_).

```
radius = 10
```
(more explicit compare to `x = 10`)

```
My_roll_no = 1025
```

```
Current_year = 2024
```

- Not allowed **digits before letters** in Variable naming

```
x9 = 10
print (x9)
```

```
10
```

```
9x = 10
print (9x)
```

```
  Cell In[2], line 1
    9x = 10
    ^
SyntaxError: invalid decimal literal
```

Contd…

Reserved keywords in Python **cannot** be used as variables

```
import = 5
print (import)
```

```
  Cell In[3], line 1
    import = 5
           ^
SyntaxError: invalid syntax
```

Special characters (@, ?, !, etc.) other than underscores are **not allowed.**

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

```
import keyword

print(keyword.kwlist)
```

```
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue', 'def', 'del', 'elif', 'else', 'except', 'fin
ally', 'for', 'from', 'global', 'if', 'import', 'in', 'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'wi
th', 'yield']
```

Multiple assignments

Python allows **multiple assignments** using

1. **Successive multiple equal signs:** the value on the far right is assigned to all variables names on the left side of an equal sign.
2. **By having several variables names and values separated by commas on both sides:** the values are respectively assigned to the variables names.

```
#Successive multiple equal signs
var1 = var2 = var3 = 99

#By having several variables names and values separated by commas on both sides:
var4, var5 = 101, 201

print(var1)
print(var2)
print(var3)
print(var4)
print(var5)
```

```
99
99
99
101
201
```

Type conversion

Changing the type of a variable

- int/float → string; using the `str()` function
- Number as a string → int/float; using the `int()` and `float()` functions

```
# Type Conversion 1
number_as_int = 5
same_number_as_string = str(number_as_int)
print(type(number_as_int))
print(type(same_number_as_string))

#print("Printing the values of the variables")

print(number_as_int)
print(same_number_as_string)


#print("Going to the next type conversion examples")
#print("\n")

number_as_string = "5"
change_to_int = int(number_as_string)
print (type(number_as_string))
print (type(change_to_int))
```

```
<class 'int'>
<class 'str'>
5
5
<class 'str'>
<class 'int'>
```

Contd…

- if the string **includes a decimal point**, it **cannot be converted to int**. Can only be converted to **float**

```
number_as_string = "5.0"
change_to_float = float(number_as_string)
print (type(number_as_string))
print (type(change_to_int))

print("Hello World !!!")

change_to_int = int(number_as_string)
print (type(number_as_string))
print (type(change_to_int))
```

```
<class 'str'>
<class 'int'>
Hello World !!!
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[22], line 8
      4 print (type(change_to_int))
      6 print("Hello World !!!")
----> 8 change_to_int = int(number_as_string)
      9 print (type(number_as_string))
     10 print (type(change_to_int))

ValueError: invalid literal for int() with base 10: '5.0'
```

- if the string contains **non-digits characters**, the conversion fails.

```
number_as_string = "5.x$"
change_to_float = float(number_as_string)
print (type(number_as_string))
print (type(change_to_int))
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
Cell In[23], line 2
      1 number_as_string = "5.x"
----> 2 change_to_float = float(number_as_string)
      3 print (type(number_as_string))
      4 print (type(change_to_int))

ValueError: could not convert string to float: '5.x'
```

Why type casting/conversion is necessary?

Input from User

Problem: Ask user to enter two numbers using keyboard and print them.

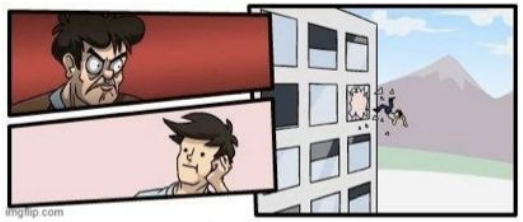
Solution: To take input from keyboard, we need to use `input()` function.

```
#input form user using keyboard

# Problem: Enter two numbers and print them.

first_number = input("Enter the first number")
second_number = input("Enter the second number")

print(first_number)
print(second_number)
```

```
 Enter the first number 100
Enter the second number ↑↓ for history. Search history with c-↑/c-↓
```

Programming test - 2

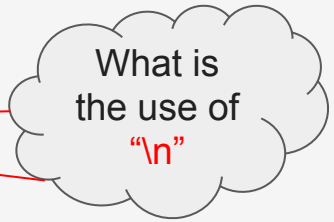
Problem: Enter a number using keyboard and print the double of the number i.e., 2 * entered number.



What's wrong with python™ ?

Revisiting type casting

```
# Type casting and user input
# Problem: Enter a number using keyboard and print the double of the number i.e., 2 * entered number.

user_input = input("Enter a number")

wrong_output = 2 * user_input
print(wrong_output)


print("\n")
print("The correct output is shown below: \n")

correct_input = 2 * int(user_input)
print(correct_input)
```




```
Enter a number 5
55


The correct output is shown below:

10
```

Note: `input()` function always returns `str` data type. We need to convert a string variable to a number (int/float) variable before doing any arithmetic operation.

Programming test - 3

Problem: Enter radius, value of pi, and calculate the volume of the sphere

```
# calculate the volume of a sphere

radius = float(input("Enter the radius: "))
my_pi = float(input("Enter the value of Pi: "))
sphere_volume = (4/3) * my_pi * radius**3
print("The volume of sphere is")
print(sphere_volume)
```

```
Enter the radius:  3
Enter the value of Pi:  3.14
The volume of sphere is
113.03999999999999
```

```
#import math

radius = float(input("Enter the radius: "))
my_pi = float(input("Enter the value of Pi: "))
sphere_volume = (4/3) * my_pi * pow(radius, 3)
print("The volume of sphere is")
print(sphere_volume)
```

```
Enter the radius:  3
Enter the value of Pi:  3.14
The volume of sphere is
113.03999999999999
```


```
The volume of sphere is: 113.0399
```

Contd…

- The **`format()`** function can be used to insert the value of a variable in another string variable.
- It requires some placeholders {} in the string, and some variables passed to the **`format()`** method.

```
1 a = 10
2 b = 8
3 c = 4
4 message = "The values of (a,b,c) are: {}, {}, and {}".format(a, b, c)
5 print(message)
```

```
The values of (a,b,c) are: 10, 8, and 4
```

```
#using the format option

radius = float(input("Enter the radius: "))
my_pi = float(input("Enter the value of Pi: "))
sphere_volume = (4/3) * my_pi * pow(radius, 3)
print("The volume of sphere is")
print(sphere_volume)

print("\n.....................................")
print(f"The volume of sphere is: {sphere_volume}, when the radius is: {radius} \n")

print(".....................................")
message = "The volume of sphere is: {}, when the radius is: {} \n".format(sphere_volume, radius)
print(message)
```

```
Enter the radius:  3
Enter the value of Pi:  3.14
The volume of sphere is
113.03999999999999

.....................................
The volume of sphere is: 113.03999999999999, when the radius is: 3.0

.....................................
The volume of sphere is: 113.03999999999999, when the radius is: 3.0
```

Quick Exercise

Write a Python script that explicitly asks the user for its name and its age.

Later on, the scripts should display a message, reading

Your name is: _____, and your age is ____.

```
'''
Write a Python script that                    r its name and its age.
Later on, the scripts shou                    g
Your name is: _________, a
'''
#ask the user's name
user_name = input("Enter y

#ask age
user_age = input("What is

#Showing the output
print(f"Your name is {user                    r_age}")
```

```
Enter your name:  ILP Course
What is your age ? 2024
Your name is ILP Course, and your age is 2024
```

Have you noticed something?

Is there any mistake?

Type cast

Block comments

- **Block comments** are comments starting and ending with **triple quotation marks**

  → `"""Your multiple lines of comments """`.

- These are great for headers in your files: specifying the author name, a brief description of what the script does, etc.

- All comments are **ignored** when **executing the code**, they only serve for description.

Programming test - 4

Problem 1: Compute the roots of a quadratic equation, with strictly positive determinant

- Consider the quadratic equation $ax^2+bx+c=0$
- With $a, b, c \in \mathbb{R}$, such that the determinant $\Delta>0$
  $\Delta=b^2-4ac$
- The two roots of this equation (x_1,x_2) are given by
  $x_1=(-b+\sqrt{\Delta})/2a \quad and \quad x_2=(-b-\sqrt{\Delta})/2a$

Task: Consider a quadratic equation with the following values: $a=2$, $b=-2$ and $c=-24$.

Write a Python program

1. that **computes** the value of **Δ** and **prints** it on screen (to check it is indeed strictly positive)
2. And, later on, **computes** the values of the roots x_1 and x_2 and **prints** them on screen.

Expected answers: Δ = 196, x_1 = 4, x_2 = -3.

Problem 2: You want to distribute 50 apples equally among 7 friends. How many apples will each friend receive, and how many apples will be left over?

Problem 3: You have a rectangular garden with a length of 12m and a width of 8 m. If you want to fence the perimeter of the garden with a fence that costs $5 per meter, how much will it cost you? (*Use arithmetic operators and logical operators*)

Problem 4: If x/3 = y/4 = z/7, then find the value of (x+y+z)/2z

What we've learnt

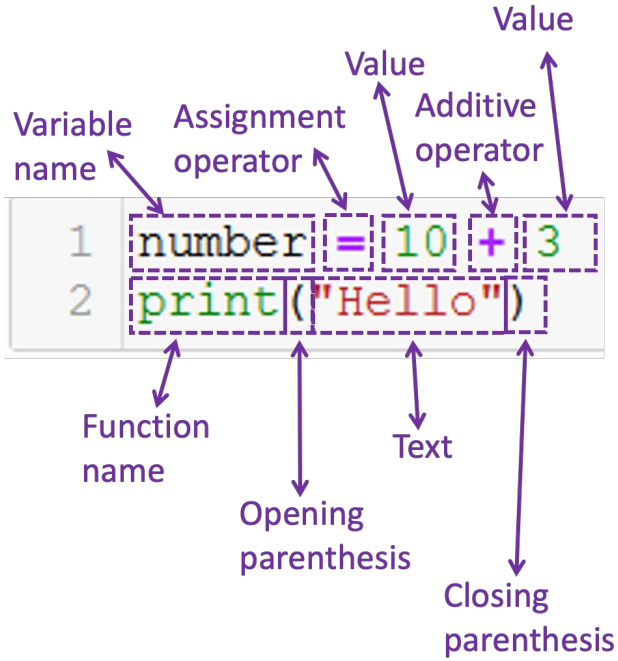
- What is Programming language?
- Why it is needed?
- Discussion on Algorithm and Logic
- How to install Python, Jupyter notebook,

  Google colab

- Basic Python syntax, different operators,

  Variable, Keywords, Type casting

- How to take input from user
-



See you in the next class !