

# Python Installation

The slides in this document are guides for installing Python on your computers.

The installation should be in the following order:

- 1. Install the latest version of Python (the engine)
- 2. Install the PyCharmIDE (a software tools that sits on top of your Python to help you work efficiently and conveniently).

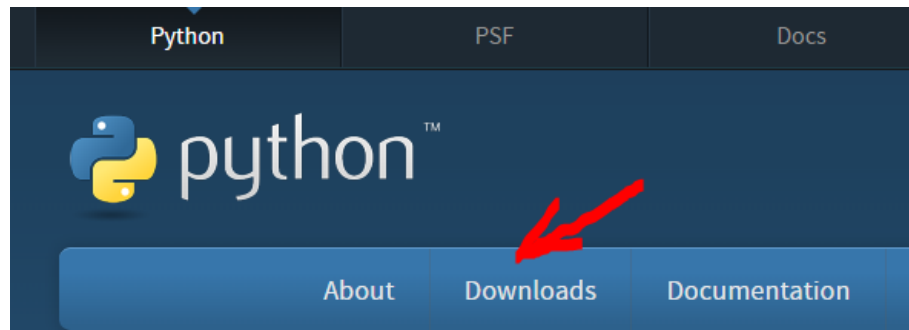
## Accessing Python Without Installation

You can use an online Python compilers as a **temporary** measure before installing, e.g.,

- ❑ <https://www.programiz.com/python-programming/online-compiler/>
- ❑ <https://www.online-python.com/>

# Installing Python and PyCharm IDE (Windows)

1. Go to <http://www.python.org> and click on "Downloads"



2. Ideally install the latest version (at the time of writing Python is on version 3.12.1). However, an earlier Python 3 will work for our practical exercises.



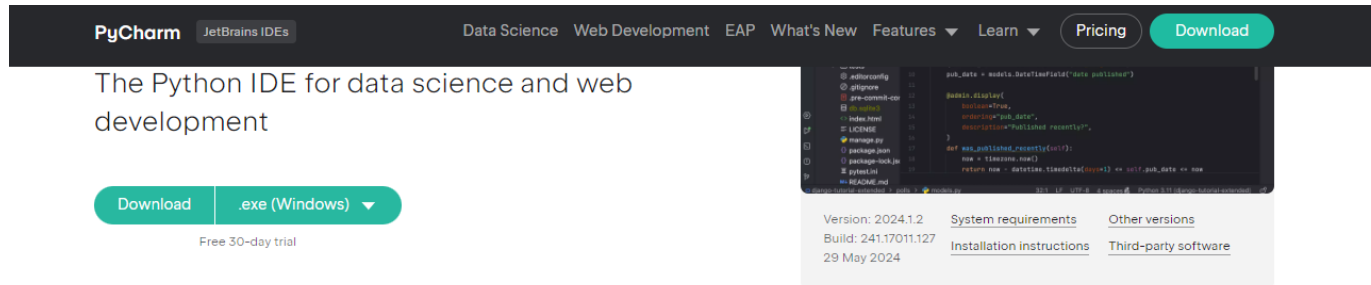
3. Install Python using the downloaded file.

# Next: Install PyCharm

- 4. When Python is installed, go to <https://www.jetbrains.com/pycharm/> and download and install the PyCharm IDE (Integrated Development Environment)



## 5. Download and Install the free (community) version.



The screenshot shows the PyCharm website's download page. At the top, there's a navigation bar with 'PyCharm', 'JetBrains IDEs', and links for 'Data Science', 'Web Development', 'EAP', 'What's New', 'Features', 'Learn', 'Pricing', and a 'Download' button. The main heading reads 'The Python IDE for data science and web development'. Below this, there's a 'Download' button and a dropdown menu set to '.exe (Windows)'. A note says 'Free 30-day trial'. To the right, there's a preview of the PyCharm IDE interface showing code. Below the preview, there's a table with links for 'System requirements', 'Installation instructions', and 'Third-party software'.

PyCharm JetBrains IDEs

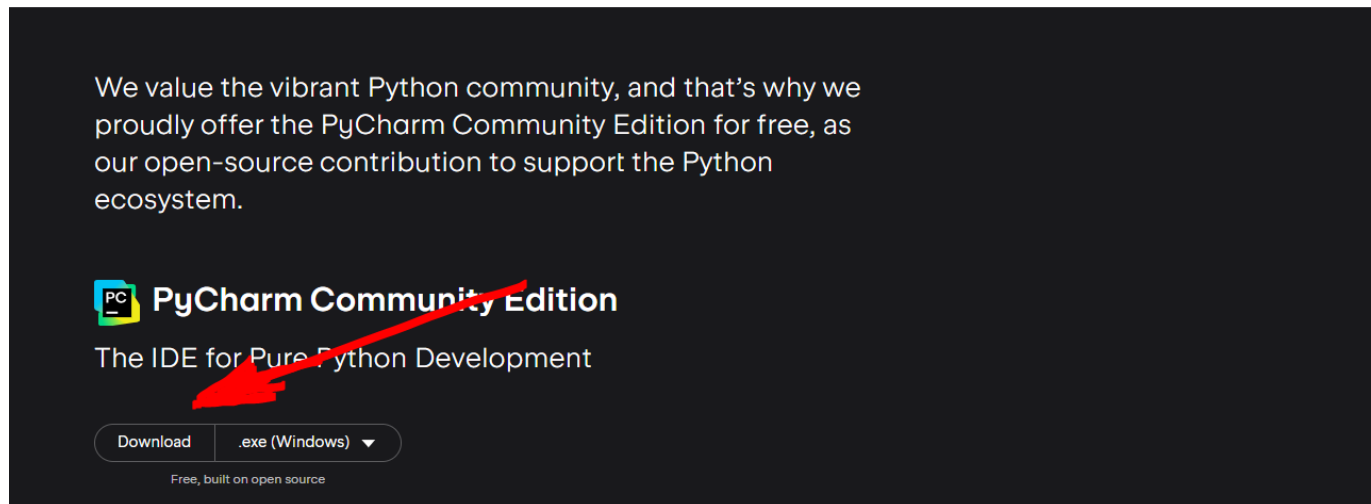
Data Science Web Development EAP What's New Features Learn Pricing Download

The Python IDE for data science and web development

Download .exe (Windows) Free 30-day trial

Version: 2024.1.2 Build: 241.17011.127 29 May 2024

System requirements Installation instructions Other versions Third-party software



The screenshot shows the PyCharm Community Edition download page. It features the PyCharm logo and the text 'PyCharm Community Edition' with a red arrow pointing to it. Below this, it says 'The IDE for Pure Python Development'. At the bottom, there's a 'Download' button and a dropdown menu set to '.exe (Windows)'. A note says 'Free, built on open source'.

We value the vibrant Python community, and that's why we proudly offer the PyCharm Community Edition for free, as our open-source contribution to support the Python ecosystem.

 **PyCharm Community Edition**

The IDE for Pure Python Development

Download .exe (Windows) Free, built on open source

# Python for Mac OS

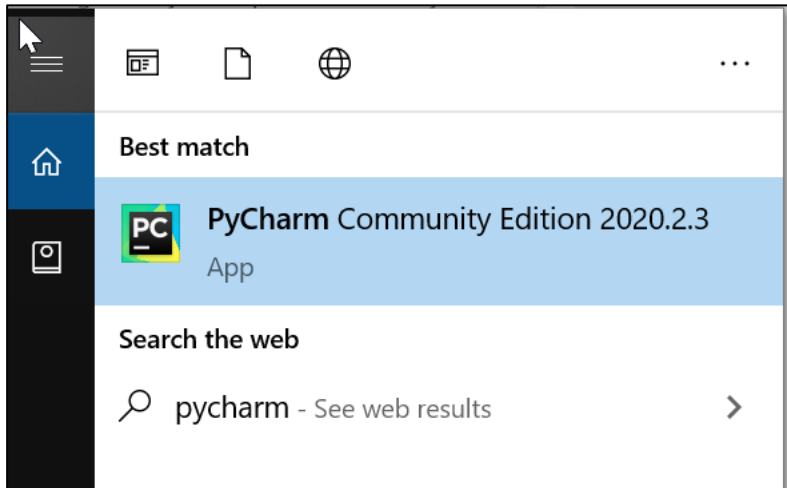
On <http://www.python.org/downloads> , click on "Mac OS X"

Download and install the latest Python 3 release:

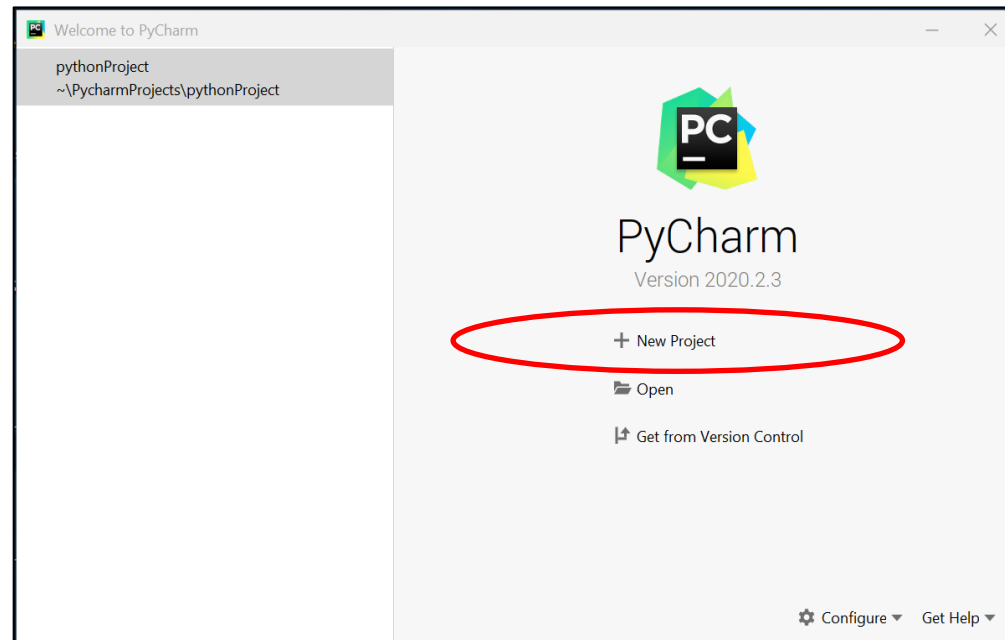


# Setting PyCharm IDE

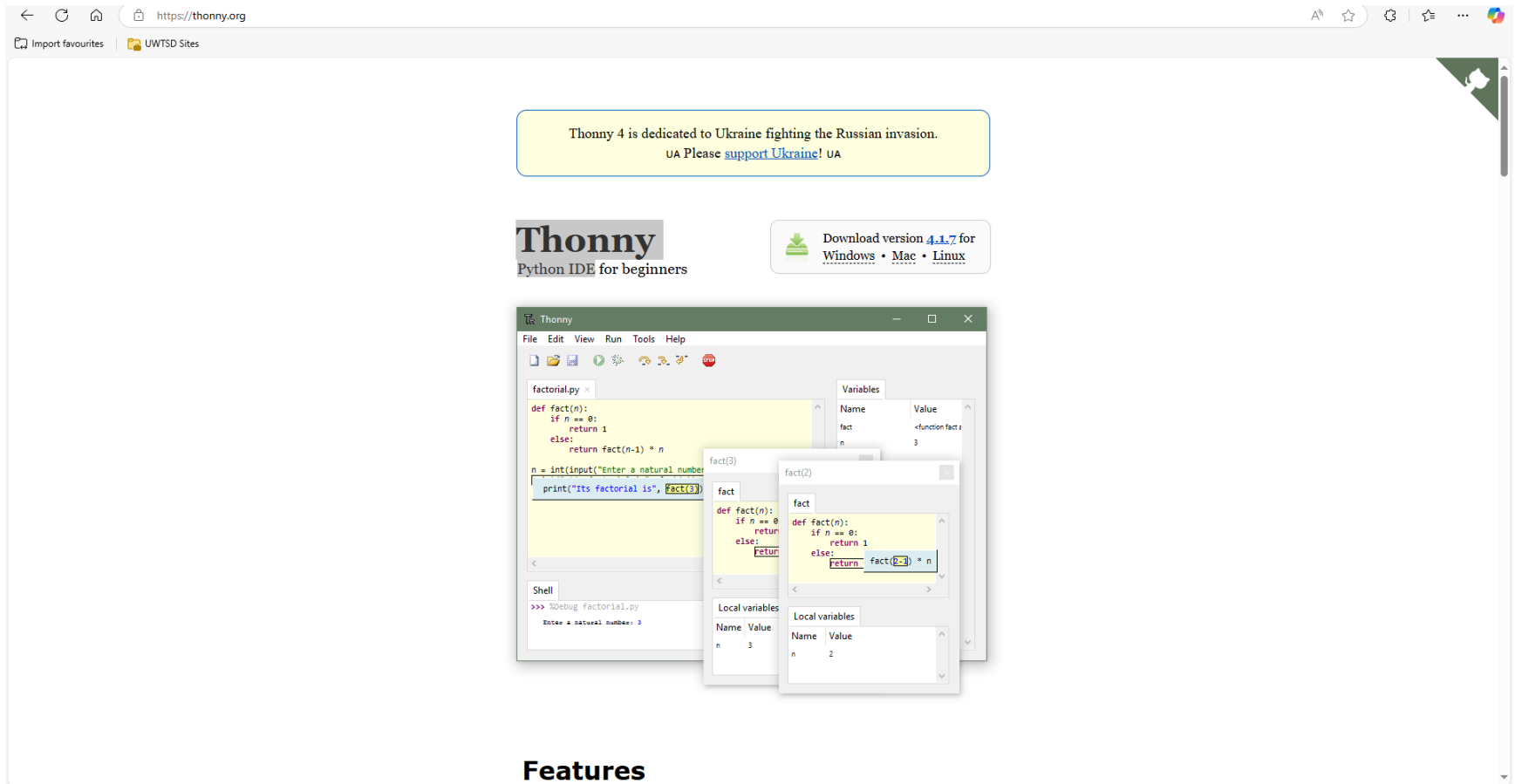
1. Start PyCharm by clicking on the shortcut on your Desktop (if you can't find it – search for "PyCharm" on your computer)



2. On first run, you should see something like this. Click on "New Project"



# Thonny Python IDE



[Thonny, Python IDE for beginners](https://thonny.org/)

<https://thonny.org/>

- For the instructions on how to set up Python Project in PyCharm, follow the instructions on the file “Setting up Python Project in PyCharm” from Moodle.

# Simple codes in Python

You can use an online Python compilers as a **temporary** measure before installing, e.g.,

- ❑ <https://www.programiz.com/python-programming/online-compiler/>
- ❑ <https://www.online-python.com/>

# Today's Session

- Basic syntax and structure of Python
- Translators
- variables
- data types
- basic input/output operations

# Recap Quiz - SDLC

A company wants to develop a new e-commerce website. The business team gathers requirements, and the development team starts coding immediately. What major SDLC phase is missing in this process?

- A) Testing
- B) Design
- C) Deployment
- D) Maintenance

# Recap Quiz - SDLC

A software development team is working on a banking application. Before coding, they gather requirements and create detailed documentation. Which SDLC phase does this belong to?

- a) Testing
- b) Maintenance
- c) Requirement Analysis
- d) Deployment

# Recap Quiz - SDLC

A software development team is building a mobile application. The customer requests frequent changes in features. What SDLC model is best suited for this scenario?

- A) Waterfall Model
- B) Spiral Model
- C) Agile Model
- D) V-Model

# Recap Quiz - SDLC

A project is behind schedule. The manager decides to assign more developers to certain tasks to complete them faster. How will this be represented in a Gantt chart?

- A) The task duration decreases
- B) The task duration increases
- C) The dependencies change automatically
- D) The tasks are removed

# Recap Quiz - SDLC

- **Which of the following is NOT an advantage of using a Gantt chart?**
  - a) Helps visualize project timelines
  - b) Clearly shows task dependencies
  - c) Automatically adjusts schedules when changes occur
  - d) Easy to track project progress

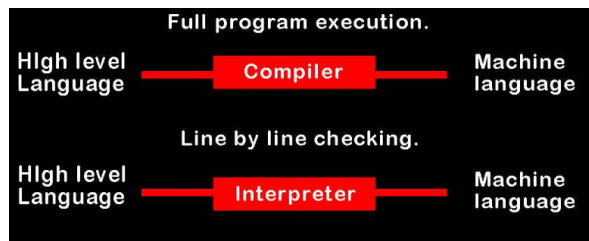
# Translators & High-level Languages

Program---→Translator--→MachineCode

## Translator

- Compiler
- Interpreter

**Thonny/pycharm - Integrated Development Environment (IDE) for Python**



## Machine Code & High-level Languages



Machine Code and High level Languages Using Interpreters and Compilers

Video:

<https://youtu.be/1OukpDfsuXE?si=Rk-e3SqmSTJrCeC0>

# Let us Python

**Task:** Write a program that asks the user for their name and age, then prints them.



```
Thonny - <untitled> @ 7:11
File Edit View Run Tools Help
<untitled>
1 # Ask for user input
2 name = input("Enter your name: ")
3 age = input("Enter your age: ")
4
5 # Display output
6 print(name)
7 print(age)
8

Shell
>>>

Local Python 3 • Thonny's Python
```

# Let us Python

**Task:** Write a program that asks the user for their name and age, then prints them.



The screenshot shows the Thonny Python IDE interface. The main editor window displays a Python script with the following code:

```
1 # Ask for user input
2 name = input("Enter your name: ")
3 age = input("Enter your age: ")
4
5 # Display output
6 print(name)
7 print(age)
8
```

Below the editor is a shell window with the prompt `>>>`.

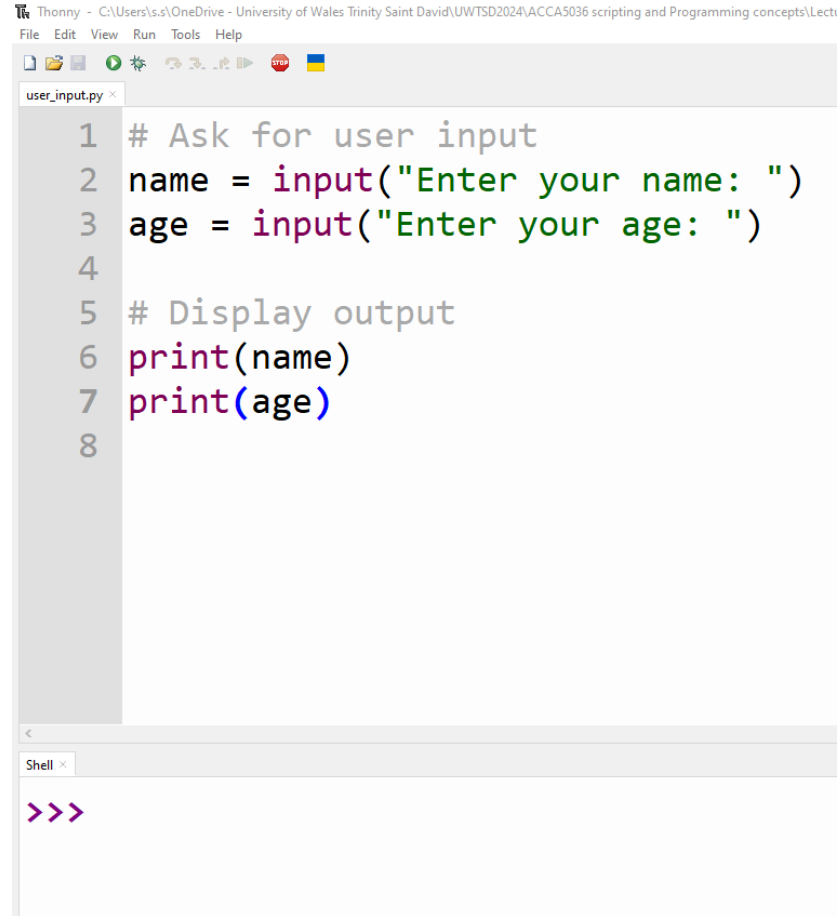
- Use lowercase letters
- Use underscores (\_) to separate words
- Keep names descriptive but concise
- Avoid using Python built-in names
- Use meaningful names for different project files
- Follow Python Enhancement Proposal 8 (PEP 8) guidelines

Now, name this file.....

# Let us Python

**Task:** Write a program that asks the user for their name and age, then prints them.

user\_input.py



```
Thonny - C:\Users\s.s\OneDrive - University of Wales Trinity Saint David\UWTSD2024\ACCA5036 scripting and Programming concepts\Lecti
File Edit View Run Tools Help
user_input.py x
1 # Ask for user input
2 name = input("Enter your name: ")
3 age = input("Enter your age: ")
4
5 # Display output
6 print(name)
7 print(age)
8

Shell x
>>>
```

# Let us Python

**Task:** Write a program that asks the user for their name and age, then prints them.

Thonny - C:\Users\s.s\OneDrive - University of Wales Trinity Saint David\UWTSD2024\ACCA5036 scripting and Programming concepts\Lecture slides\user\_input.py @ 8 : 1

File Edit View Run Tools Help



user\_input.py x

```
1 # Ask for user input
2 name = input("Enter your name: ")
3 age = input("Enter your age: ")
4
5 # Display output
6 print(name)
7 print(age)
8
```



<

Shell x

>>> |

# Let us Python

**Task:** Write a program that asks the user for their name, age, and favourite color, then prints them.

# Let us Python

**Task:** Write a program that asks the user for their name, age, and favourite color, then prints them.

```
Thonny - C:\Users\s.s\OneDrive - University of Wales Trinity Saint David\UWTSD2024\ACCA5036 scripting and Programming concepts\Lecture slides\user_input.py @ 9
File Edit View Run Tools Help

user_input.py x
1 # Ask for user input
2 name = input("Enter your name: ")
3 age = input("Enter your age: ")
4 color = input("enter your fav color: ")
5 # Display output
6 print(name)
7 print(age)
8 print(color)
9
```

# Let us Python

**Task:** Write a program that asks the user for their name, age, and favourite color, then prints them. Let's make it user friendly

```
# Ask for user input
name = input("Enter your name: ")
age = input("Enter your age: ")
color = input("Enter your favorite color: ")
```

```
# Display output
print(f"Hello, {name}! You are {age} years old and {color} is your favourite color.")
```

placeholders that  
will be replaced  
by the values of  
the variables

**f-string** (formatted string literal)

# String Data Type - Python

Used to store text enclosed in single (') or double (") quotes.

```
name = "Alice"  
message = 'Hello, World!'  
Print(type(name))
```

Assignment operation

# String Data Type - Python

Used to store text enclosed in single (') or double (") quotes.

```
text = "Python" } Assignment operation  
print(text[0])  
print(text[1:4])  
print(text.upper())
```

# String Data Type - Python

Used to store text enclosed in single (') or double (") quotes.

```
text = "Python" } Assignment operation  
print(text[0])  
print(text[1:4])  
print(text.upper())
```

A diagram illustrating string indexing for the word "Python". A horizontal table shows the characters and their corresponding indices. A blue callout box labeled "Starting index (inclusive)" points to index 1 (character 'y'). Another blue callout box labeled "End index (exclusive)" points to index 4 (character 'o'). A purple bracket above the table spans from index 1 to index 4, representing the slice text[1:4].

|           |   |   |   |   |   |   |
|-----------|---|---|---|---|---|---|
| Character | P | y | t | h | o | n |
| Index     | 0 | 1 | 2 | 3 | 4 | 5 |

# String Data Type - Python

Used to store text enclosed in single (') or double (") quotes.

```
text = "Python" }
```

Assignment operation

```
print(text[0])
```

```
print(text[1:4])
```

```
print(text.upper())
```

```
print(text[0:3])
```

```
print(text[:4])
```

```
print(text[2:])
```

```
print(text[-3:])
```

```
print(text[::-2])
```

| Character | P | y | t | h | o | n |
|-----------|---|---|---|---|---|---|
| Index     | 0 | 1 | 2 | 3 | 4 | 5 |

# String Data Type - Python

Used to store text enclosed in single (') or double (") quotes.

```
text = "Python" }
```

Assignment operation

```
print(text[0])
```

```
print(text[1:4])
```

```
print(text.upper())
```

```
print(text[0:3])
```

```
print(text[:4])
```

```
print(text[2:])
```

```
print(text[-3:])
```

```
print(text[::-2])
```

| Character | P | y | t | h | o | n |
|-----------|---|---|---|---|---|---|
| Index     | 0 | 1 | 2 | 3 | 4 | 5 |

# Numeric Datatypes - Python

Used for storing numbers.

```
a1 = "5"
```

```
a2 = 5
```

```
b = 3.14
```

```
c = 2 + 5j
```

```
print(type(a1))
```

```
print(type(a2))
```

```
print(type(b))
```

```
print(type(c))
```



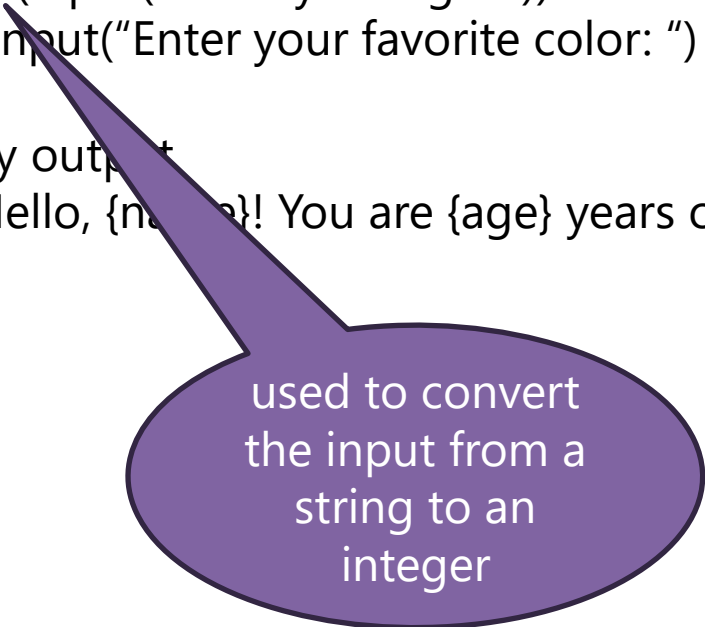
Assignment operation

# Let us Python

**Task:** Write a program that asks the user for their name, age, and favourite color, then prints them. Let's make it user friendly

```
# Ask for user input
name = input("Enter your name: ")
age = int(input("Enter your age: "))
color = input("Enter your favorite color: ")
```

```
# Display output
print(f"Hello, {name}! You are {age} years old and {color} is your favourite color.")
```



used to convert  
the input from a  
string to an  
integer

# Let us Python

**Task:** Write a program that asks the user for num1 and num2 then prints the result of adding num1 + num2. Make it user friendly

```
# User input
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

# Perform addition
result = num1 + num2

# Display the result
print("Result:", result)
```

A **test case** is a set of conditions or variables used to determine if a system or software program is working correctly.

# Let us Python

**Task:** Write a program that asks the user for num1 and num2 then prints the result of adding num1 + num2. Make it user friendly

Test cases:

# User input

```
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))
```

# Perform addition

```
result = num1 + num2
```

# Display the result

```
print("Result:", result)
```

| Test Case # | Input (num1) | Input (num2) | Expected Output    |
|-------------|--------------|--------------|--------------------|
| 1           | 5            | 3            | Result: 8          |
| 2           | -5           | 3            | Result: -2         |
| 3           | 0            | 0            | Result: 0          |
| 4           | -7           | -3           | Result: -10        |
| 5           | 10           | 100          | Result: 110        |
| 6           | 999          | 1            | Result: 1000       |
| 7           | 123456789    | 987654321    | Result: 1111111110 |
| 8           | 0            | -100         | Result: -100       |
| 9           | -1           | 1            | Result: 0          |

# Let us Python

**Task:** Write a program that asks the user for num1 and num2 then prints the result of perform the respective arithmetic operation: addition, subtraction, multiplication, or division of num1 and num2. Make it user friendly

And identify the Test cases to test the program.

```
# User input
num1 = float(input("Enter first number: "))
num2 = float(input("Enter second number: "))

# Perform addition
result = num1 + num2

# Display the result
print("Result:", result)
```

# Let us Python

**Task:** Write a program that asks the user for num1 and num2 then prints the result of perform the respective arithmetic operation: addition, subtraction, multiplication, or division of num1 and num2. Make it user friendly

And identify the Test cases to test the program.

```
# User input
num1 = int(input("Enter first number: "))
num2 = int(input("Enter second number: "))

# Perform addition
result_addition = num1 + num2
result_subtraction = num1 - num2
result_multiplication = num1 * num2
result_division = num1 / num2
# Display the result
print("Sum:", result_addition)
print("Difference:", result_subtraction)
print("Product:", result_multiplication)
print("Quotient:", result_division)
```

## Test cases

| Test Case # | Input (num1, num2) | Expected Sum | Expected Difference | Expected Product | Expected Quotient         | Description                                      |
|-------------|--------------------|--------------|---------------------|------------------|---------------------------|--------------------------------------------------|
| 1           | 10, 5              | 15           | 5                   | 50               | 2.0                       | Basic case with two positive numbers.            |
| 2           | -10, 5             | -5           | -15                 | -50              | -2.0                      | Case with a negative and a positive number.      |
| 3           | 10, -5             | 5            | 15                  | -50              | -2.0                      | Case with a positive and a negative number.      |
| 4           | -10, -5            | -15          | -5                  | 50               | 2.0                       | Both numbers are negative.                       |
| 5           | 0, 5               | 5            | -5                  | 0                | 0.0                       | First number is zero.                            |
| 6           | 5, 0               | 5            | 5                   | 0                | Error (ZeroDivisionError) | Second number is zero (division by zero error).  |
| 7           | 0, 0               | 0            | 0                   | 0                | Error (ZeroDivisionError) | Both numbers are zero, causing division by zero. |
| 8           | 100, 25            | 125          | 75                  | 2500             | 4.0                       | Large positive values.                           |
| 9           | 1, 3               | 4            | -2                  | 3                | 0.3333 (approx.)          | Testing decimal result in division.              |

# Activities

## Exercise 1: Age in Future

📌 **Task:** Ask the user for their current age and calculate how old they will be in 10 years.

◆ **Example Output:**

Enter your current age: 25 In 10 years, you will be 35 years old.

**Task:** Convert temperature from Celsius to Fahrenheit.

◆ **Formula:**  $F = (C \times 9/5) + 32$

◆ **Example Output:**

Enter temperature in Celsius: 25 Temperature in Fahrenheit: 77.0°F

# Understanding Variables in Python

```
# Assign values to variables
```

```
name = "John"
```

```
age = 25
```

```
favorite_color = "blue"
```

```
# Print a sentence using the variables
```

```
print("My name is", name, "I am", age, "years old and my favorite color is", favorite_color)
```

- Change the values and rerun the code!
- Use **f-strings** instead of concatenation.

# Understanding Variables in Python

```
# Assign values to variables
```

```
name = "John"
```

```
age = 25
```

```
favorite_color = "blue"
```

```
# Print a sentence using the variables
```

```
print("My name is", name, "I am", age, "years old and my favorite color is", favorite_color)
```

- Change the values and rerun the code!
- Use **f-strings** instead of concatenation.

```
print(f"My name is {name}, I am {age} years old and my favorite color is {favorite_color}.")
```

# Working with Data Types

- Create a variable for each data type: string, integer, float, and boolean.
- Print each variable along with its type using the type() function.

```
# Define variables with different data types
```

```
text = "Hello, Python!"    # String  
number = 10                # Integer  
decimal = 3.14             # Float  
is_python_fun = True       # Boolean
```

```
# Print values and their types
```

```
print(type(text), text)  
print(type(number), number)  
print(type(decimal), decimal)  
print(type(is_python_fun), is_python_fun)
```

# Working with Data Types

- Convert an integer to a float and a float to an integer.
- Convert a boolean to a string.

```
# Define variables with different data types

text = "Hello, Python!"    # String
number = 10                # Integer
decimal = 3.14             # Float
is_python_fun = True       # Boolean

# Print values and their types
print(type(text), text)
print(type(number), number)
print(type(decimal), decimal)
print(type(is_python_fun), is_python_fun)
```

# Working with Data Types

- Convert an integer to a float and a float to an integer.
- Convert a boolean to a string.

```
# Define variables with different data types
```

```
text = "Hello, Python!"   # String  
number = 10               # Integer  
decimal = 3.14            # Float  
is_python_fun = True      # Boolean
```

```
# Print values and their types
```

```
print(type(text), text)  
print(type(number), number)  
print(type(decimal), decimal)  
print(type(is_python_fun), is_python_fun)
```

```
print(float(number))   # Convert integer to float  
print(int(decimal))    # Convert float to integer  
print(str(is_python_fun)) # Convert boolean to string
```

# Mini Project - Simple Profile Generator

- Objective:

Apply knowledge of variables, data types, and input/output to build a simple program.

- Instructions:

1. Ask the user for their name, age, favorite hobby, and country.
2. Store the inputs in variables and print a small profile.

# Mini Project - Simple Profile Generator

- Objective:

Apply knowledge of variables, data types, and input/output to build a simple program.

- Instructions:

1. Ask the user for their name, age, favorite hobby, and country.
2. Store the inputs in variables and print a small profile.

```
Enter your name: Emma
Enter your age: 22
Enter your favorite hobby: Painting
Enter your country: Canada
```

```
=== User Profile ===
```

```
Name: Emma
```

```
Age: 22
```

```
Hobby: Painting
```

```
Country: Canada
```

```
Nice to meet you!
```

# Mini Project - Simple Profile Generator



- Objective:

Apply knowledge of variables, data types, and input/output to build a simple program.

- Instructions:

1. Ask the user for their name, age, favorite hobby, and country.
2. Store the inputs in variables and print a small profile.

```
# Collect user details
```

```
name = input("Enter your name: ")
```

```
age = int(input("Enter your age: ")) # Convert to integer
```

```
hobby = input("Enter your favorite hobby: ")
```

```
country = input("Enter your country: ")
```

```
# Print user profile
```

```
print("\n=== User Profile ===")
```

```
print(f>Name: {name}")
```

```
print(f>Age: {age}")
```

```
print(f>Hobby: {hobby}")
```

```
print(f>Country: {country}")
```

```
print("\nNice to meet you!")
```

Modify the program to **calculate** how many years the user has until they turn 100.

# Debugging – Fix the code

- Objective:

Identify and fix incorrect variable declarations.

```
my-name = "Emma"  
2age = 25  
favorite food = "Pizza"  
isPythonFun = "yes"  
  
print(My-name, "is", 2age, "years old and loves", favorite food)
```

# Debugging – Fixed code

- Objective:

Identify and fix incorrect variable declarations.

```
my_name = "Emma"  
age2 = 25  
favorite_food = "Pizza"  
is_python_fun = "yes"  
  
print(my_name, "is", age2, "years old and loves", favorite_food)
```

# Debugging – Fix the code

- Objective:

Fix errors related to data types and type conversion.

```
length = "50"  
width = 30.5  
area = length * width # Calculate area  
  
print("The area of the rectangle is " + area)
```

# Debugging – Fixed code

- Objective:

Fix errors related to data types and type conversion.

```
length = int("50")  # Convert to integer
width = 30.5
area = length * width

print(f"The area of the rectangle is {area}")
```

# Debugging – Fix the code

- Objective:

Fix errors related to taking input from users.

```
name = input(Enter your name: )  
age = input("Enter your age: ")  
print("Hello, " name "! You are" age "years old.")
```

# Debugging – Fixed code

- Objective:

Fix errors related to taking input from users.

```
name = input("Enter your name: ")
age = input("Enter your age: ")

print("Hello, " + name + "! You are " + age + " years old.")
```

# Debugging – Fix the code

- Objective:

Fix incorrect mathematical operations.

```
num1 = input("Enter first number: ")  
num2 = input("Enter second number: ")  
sum = num1 + num2  
print("The sum of the numbers is:", sum)
```

# Debugging – Fixed code

- Objective:

Fix incorrect mathematical operations.

```
num1 = int(input("Enter first number: "))  
num2 = int(input("Enter second number: "))  
total = num1 + num2  
  
print("The sum of the numbers is:", total)
```

# Debugging – Fix the code

- Objective:

Fix errors related to printing formatted output.

```
name = input("Enter your name: ")
age = input("Enter your age: ")

print("Hello {name}, you are {age} years old!")
```

# Debugging – Fixed code

- Objective:

Fix errors related to printing formatted output.

```
name = input("Enter your name: ")
age = input("Enter your age: ")

print("Hello {name}, you are {age} years old!")
```

```
print(f"Hello {name}, you are {age} years old!")
```

Modify the program to **ask for their city** and include it in the output.

# Debugging – Fix the code

```
name = input("Enter your name: ")
age = input("Enter your age: ")
hobby = input("Enter your hobby: ")

years_until_100 = 100 - age
print(f"{name}, you will turn 100 in {years_until_100} years!")
```

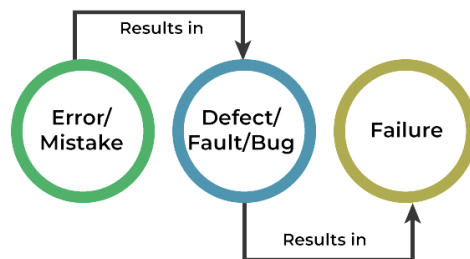
Can you find the bug?

# Debugging – Errors & Bugs

An **error** is a **mistake** in the code that prevents it from running or producing the correct output.

Errors are usually **immediately detected** by the **compiler or interpreter** (e.g., syntax errors).

```
Print("Name:", name)
```



A **bug** is a **fault or flaw** in a program that causes unexpected or incorrect behavior.

Bugs may remain **undetected** until tested or used by a user.

```
print("The sum is:", 2 * 3)
```

# Activity

Create a program that **generates a username** based on a user's **first name, last name, and birth year** using **string slicing**.

## Steps to Follow:

1. Ask the user to input their **first name, last name, and birth year**.
2. Generate a **username** using the following format:
  1. First **three letters** of the first name.
  2. First **three letters** of the last name.
  3. Last **two digits** of the birth year.
3. Display the generated username.

**Username  
Generator!**

Enter your first name: John

Enter your last name: Doe

Enter your birth year (YYYY): 1995

Your generated username is: JohDoe95

# Activity - Solution

Create a program that **generates a username** based on a user's **first name, last name, and birth year** using **string slicing**.

## Steps to Follow:

1. Ask the user to input their **first name, last name**, and **birth year**.
2. Generate a **username** using the following format:
  1. First **three letters** of the first name.
  2. First **three letters** of the last name.
  3. Last **two digits** of the birth year.
3. Display the generated username.



*# Step 1: Get user input*

```
first_name = input("Enter your first name: ")
```

```
last_name = input("Enter your last name: ")
```

```
birth_year = input("Enter your birth year (YYYY): ")
```

*# Step 2: Apply slicing*

```
username = first_name[:3] + last_name[:3] + birth_year[-2:]
```

*# Step 3: Display the generated username*

```
print("Your generated username is:", username)
```

# Activity – Research and modify the code

## Modify the project to ensure:

- The first name and last name are converted to **lowercase**. (.lower())
- The username is **always at least 6 characters long**.
- Allow the user to **customize** the username format (e.g., choose the length of slices).

```
Enter your first name: John
Enter your last name: Doe
Enter your birth year (YYYY): 1995
```

✓ Your generated username is: johndoe95

```
Enter your first name: A
Enter your last name: Z
Enter your birth year (YYYY): 2010
```

✓ Your generated username is: axzxx10

# Activity – Restaurant menu

Ask the user for their name and location

Ask them to select a dish and quantity from the menu  
(3 options).

Display the name and “your order is getting ready”

Enter your name: Alice

Welcome Alice to Foodie's Delight

Menu:

1. Burger - £5

2. Pizza - £8

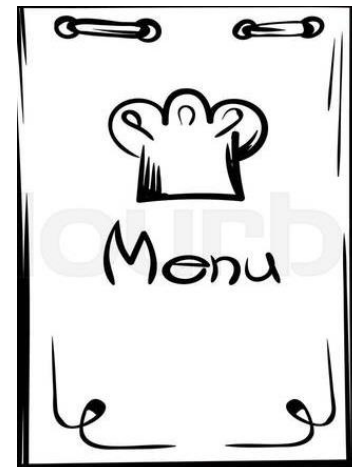
3. Pasta - £7

Enter the dish name (1/2/3): 2

Enter quantity: 2

Alice your order is getting ready

Thank you for ordering



## Next session

- More Data types and Operators
- If statements