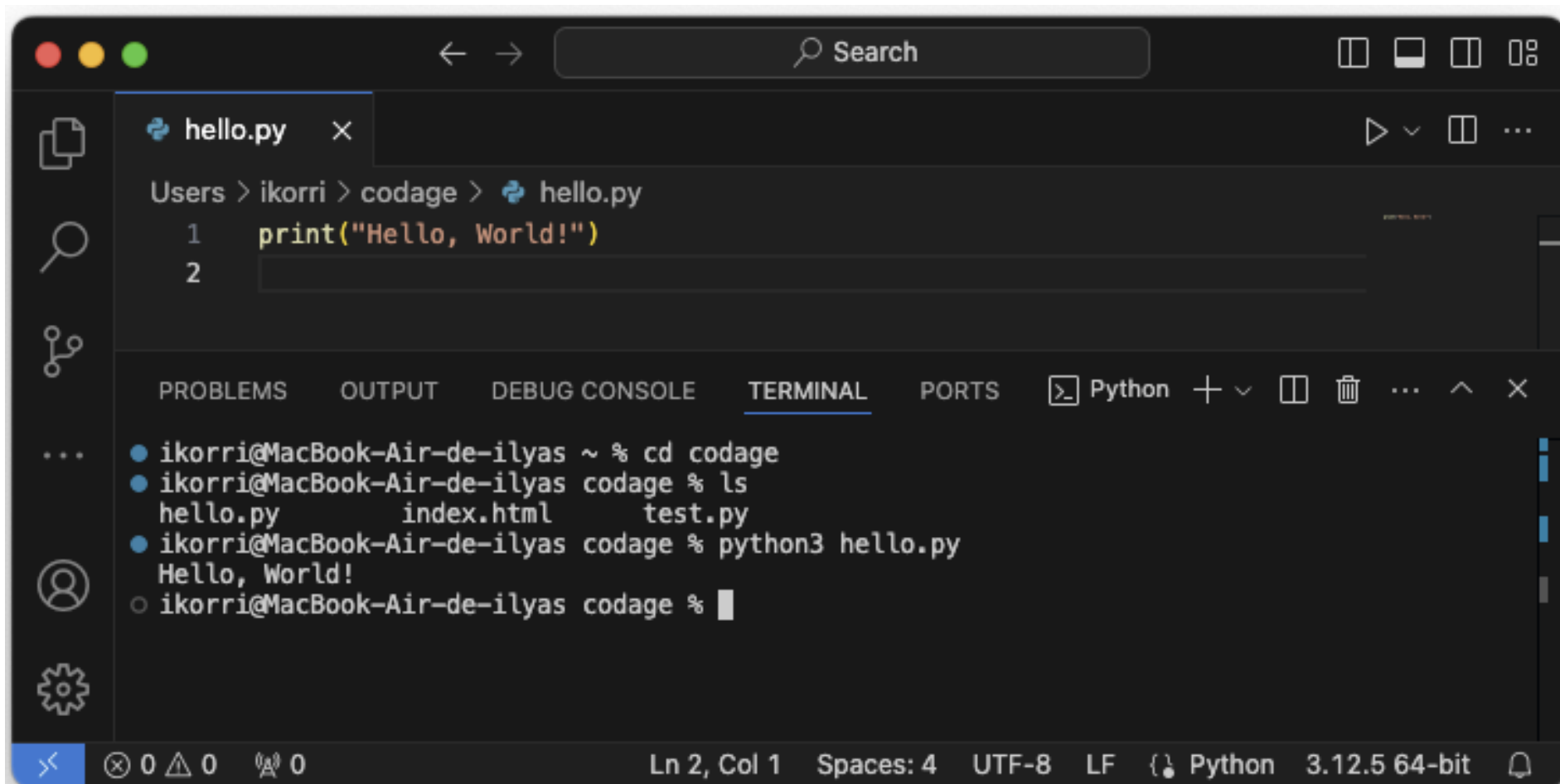


Votre Premier Programme

A screenshot of a code editor interface. The top part shows a file named 'hello.py' with the following code:

```
1 print("Hello, World!")
2
```

The bottom part shows a terminal window with the following commands and output:

```
ikorri@MacBook-Air-de-ilyas ~ % cd codage
ikorri@MacBook-Air-de-ilyas codage % ls
hello.py      index.html    test.py
ikorri@MacBook-Air-de-ilyas codage % python3 hello.py
Hello, World!
ikorri@MacBook-Air-de-ilyas codage %
```

The status bar at the bottom indicates the file is 'hello.py', 3.12.5 64-bit, and the cursor is at line 2, column 1.

Les variables



- Les variables sont des conteneurs pour stocker des données.
- En Python, vous n'avez pas besoin de déclarer le type de variable, il est automatiquement détecté.

```
nom = "KORRI"    # Chaîne de caractères  
age = 19         # Entier  
taille = 1.78    # Nombre à virgule flottante  
majeur = True    # Booléen
```

Les bonnes pratiques



Noms explicites :

Utilisez des noms de variables qui décrivent clairement leur rôle.

Mauvais : n, x

Bon : nombre_etudiants, adresse_serveur

Utilisez le format snake_case :

Séparez les mots avec un underscore _.

Exemple : **nombre_total**, **prix_unitaire**

Python possède des mots-clés réservés qui ne peuvent pas être utilisés comme noms de variables.

Exemples : if, else, while, for, True, False.



Les types de base

- **int** : Entiers (ex : 1, 2, 3)
- **float** : Nombres à virgule flottante (ex : 1.5, 3.14)
- **str** : Chaînes de caractères (ex : "Python")
- **bool** : Booléens (True ou False)

Vous pouvez connaître le type de la variable avec la fonction **type()**

Exemple:

```
nom = "KORRI" # Chaîne de caractères  
print(type(nom))
```

Les opérateurs



```
a = 10
b = 3
print(a + b)  # Addition: 13
print(a - b)  # Soustraction: 7
print(a * b)  # Multiplication: 30
print(a / b)  # Division: 3.333...
print(a // b) # Division entière: 3
print(a % b)  # Modulo: 1
print(a ** b) # Puissance: 1000
```

Les opérateurs de comparaison



```
x = 5
y = 10
print(x > y)  # False
print(x < y)  # True
print(x >= y) # False
print(x <= y) # True
print(x == y) # False
print(x != y) # True
```



Affectation combinée

- **Addition** : $x += 5$
Équivaut à : $x = x + 5$
- **Soustraction** : $x -= 3$
Équivaut à : $x = x - 3$
- **Multiplication** : $x *= 2$
Équivaut à : $x = x * 2$
- **Division** : $x /= 4$
Équivaut à : $x = x / 4$

Structure de données



- Les Listes (list)

Définition : Collection **ordonnée**, **modifiable**, et **indexée**.

```
connexions = ["SSH", "HTTP", "HTTPS"]
```

- Les Tuples (tuple)

Définition : Collection **ordonnée** et **immuable**).

```
connexion_reseau = ("192.168.1.100", 443, "HTTPS")
```

- Les Dictionnaires (dict)

Définition : Collection non ordonnée d'éléments sous forme de **paires clé-valeur**.

```
etudiant = {"nom": "Alice", "age": 21}
```


Exemple d'utilisation des variables



```
1  # Déclaration et affectation de variables
2  nom = "Alice"           # Chaîne de caractères
3  age = 25                # Entier
4  taille = 1.68           # Nombre à virgule flottante
5  est_étudiant = True     # Booléen
6
7  # Affichage des valeurs des variables
8  print("Nom:", nom)
9  print("Âge:", age)
10 print("Taille:", taille, "m")
11 print("Est étudiant:", est_étudiant)
12
13 # Modification des variables avec des opérateurs d'affectation
14 age += 1                 # Incrémente l'âge de 1 an
15 taille *= 1.05           # Augmente la taille de 5%
16 est_étudiant = not est_étudiant # Inverse le statut d'étudiant
17
18 # Affichage des valeurs après modification
19 print("\nAprès modification :")
20 print("Nom:", nom)
21 print("Âge:", age)
22 print("Taille:", taille, "m")
23 print("Est étudiant:", est_étudiant)
```