المتغيرات (Variables) وأنواع البيانات البسيطة (Simple Data Types) في بايثون

المتغيرات في بايثون هي أسماء تعطيها لقيم لتخزينها في الذاكرة. يمكن أن تكون هذه القيم من أنواع بيانات مختلفة. الأنواع البسيطة من البيانات تشمل:

1. **الأعداد الصحيحة (Integers) :** تستخدم لتخزين الأعداد الصحيحة (مثل: 1، -5، 100).
2. **الأعداد العشرية (Floating-Point Numbers) :** تستخدم لتخزين الأعداد ذات الفاصلة العشرية (مثل: 3.14، -0.5).
3. **النصوص (Strings) :** تستخدم لتخزين سلاسل من الأحرف (مثل : "Hello, World!").
4. **القيم المنطقية (Booleans) :** تستخدم لتخزين قيمتين فقط: `True` أو `False`.

1. الأعداد الصحيحة (Integers)

```
x = 5
y = -10
z = 123456789

print(x)  # Output: 5
print(y)  # Output: -10
print(z)  # Output: 123456789
```

2. الأعداد العشرية (Floating-Point Numbers)

```
pi = 3.14159
negative_decimal = -0.75

print(pi)              # Output: 3.14159
print(negative_decimal) # Output: -0.75
```

3. النصوص (Strings)

```
greeting = "Hello, World!"
name = 'Alice'

print(greeting) # Output: Hello, World!
print(name)     # Output: Alice
```

الجمع بين السلاسل النصية Combining or Concatenating Strings

يمكنك دمج سلاسل النصوص باستخدام علامة الجمع (+) .

Example:

```
first_name = "ada"

last_name = "lovelace"

full_name = first_name + " " + last_name

print("Hello, " + full_name.title() + "!")
```

Result: **"Hello, Ada Lovelace!"# Hello = النص الثابت**

4 .القيم المنطقية (Booleans)

```
is_student = True
is_teacher = False

print(is_student) # Output: True
print(is_teacher) # Output: False
```

ملاحظات إضافية:

- يمكن استخدام دالة type() لمعرفة نوع البيانات الخاص بأي متغير.

```
a = 5
b = 3.14
c = "Hello"
d = True

print(type(a))  # Output: <class 'int'>
print(type(b))  # Output: <class 'float'>
print(type(c))  # Output: <class 'str'>
print(type(d))  # Output: <class 'bool'>
```

قواعد تسمية المتغيرات في بايثون (Naming and Using Variables)

عند تسمية المتغيرات في بايثون، هناك بعض القواعد والمعايير التي يجب اتباعها لضمان أن تكون الأسماء صالحة ومفهومة للقراء. هذه القواعد تساعد على كتابة كود نظيف وقابل للقراءة والصيانة.

- يجب أن تحتوي أسماء المتغيرات على أحرف أو أرقام أو شرطات سفلية (_) فقط.
- يجب ألا تبدأ الأسماء برقم.
- يجب تجنب استخدام الكلمات المحجوزة في بايثون مثل print, if, else، إلخ.
- يجب أن تكون الأسماء واضحة ومعبرة.

Examples:

```
# أسماء صحيحة للمتغيرات
user_age = 25          # snake_case: ستخدم الحروف الصغيرة مع الفاصل السفلي _ للفصل بين الكلمات
userName = "Alice"   # camelCase: تبدأ الكلمة الأولى بحرف صغير وكل كلمة تالية تبدأ بحرف كبير
total_price = 99.99
is_student = True
_country = "Egypt"

# أسماء غير صحيحة للمتغيرات
2nd_user = "John"       # يبدأ برقم، خطأ
user-name = "Alice"     # يحتوي على رمز غير مسموح، خطأ
class = "Math"          # كلمة محجوزة، خطأ
```

بعض الإرشادات الإضافية:

- **استخدم _لتجنب التداخل مع الكلمات المحجوزة :**إذا كان اسم المتغير المراد استخدامه يتعارض مع كلمة محجوزة، يمكن استخدام (_) في نهاية الاسم.
  مثال: class_ بدلاً من class
- **تجنب الأسماء القصيرة جداً :**مثل a, b، إلا في الحلقات أو إذا كان لها معنى واضح.

تجنب أخطاء الأسماء عند استخدام المتغيرات Avoiding Name Errors When Using Variables

أخطاء الأسماء تحدث عند محاولة استخدام متغير لم يتم تعريفه مسبقًا أو إذا كان هناك خطأ إملائي في الاسم.

Example:

```
#تعريف متغير
"!message = "Hello Python Crash Course reader
#خطأ إملائي في اسم المتغير عند الطباعة
print(mesage) # خطأ:  NameError: name 'mesage' is not defined
# الحل الصحيح
print(message)#يطبع القيمة بنجاح
```

تغيير حالة الأحرف في السلسلة النصية باستخدام الدوال Changing Case in a String with Methods

بايثون توفر دوال مدمجة لتغيير حالة الأحرف في السلسلة النصية، مثل:

- **title()** للجعل الحرف الأول من كل كلمة حرف كبير.
- **upper()** لتحويل جميع الأحرف إلى حروف كبيرة.
- **lower()** لتحويل جميع الأحرف إلى حروف صغيرة.

Example:

```
name = "ada lovelace"
print(name.title())  # Result: Ada Lovelace
print(name.upper())  # Result: ADA LOVELACE
print(name.lower())  # Result: ada lovelace
```

إضافة مسافات بيضاء إلى السلاسل النصية باستخدام علامات التبويب أو الأسطر الجديدة Adding Whitespace to Strings with Tabs or Newlines

\n\t(سطر جديد) و **\n** إضافة المسافات البيضاء إلى السلاسل النصية باستخدام المحارف الخاصة مثل (تبويب) هو وسيلة لتنظيم النص وتنسيقه بشكل أفضل عند الطباعة.

Examples:

1- (الانتقال الى سطر جديد) إضافة أسطر جديدة باستخدام \n .

```
print("Languages:\nPython\nC\nJavaScript")
```

Result:

```
Languages:
Python
C
JavaScript
```

2-إضافة مسافات بادئة (مثل اضافة Tab) باستخدام \n\t .

```
print("Languages:\n\tPython\n\tC\n\tJavaScript")
```

Result:

Languages:

Python

C

JavaScript

3- تنفيذ الحالتين اعلاه معا عن طريق استخدام الايعازين \n\t و \n .

```
print("Favorite Languages:\n\tPython\n\tJava\n\tJavaScript\nGo\nRust")
```

Result:

Favorite Languages:

Python

Java

JavaScript

Go

Rust

في هذا المثال، "Go" و "Rust" ستتم طباعتهما بدون مسافة بادئة، لأن \t لم يتم استخدامه قبلهما.

إزالة المسافات البيضاء Stripping Whitespace

لإزالة المسافات البيضاء غير الضرورية من النص، يمكنك استخدام:

- **rstrip()**: لإزالة المسافات من نهاية النص.
- **lstrip()**: لإزالة المسافات من بداية النص.
- **strip()**: لإزالة المسافات من الجانبين.

Example:

```
favorite_language = ' python '
print(favorite_language.rstrip())  # يزيل المسافات البيضاء من النهاية ويطبع :  ' python'
print(favorite_language.lstrip())  # يزيل المسافات البيضاء من النهاية ويطبع :  'python '
print(favorite_language.strip())   # يزيل المسافات البيضاء من النهاية ويطبع :  'python'
```

Avoiding Type Errors with the str() Function تجنب أخطاء الأنواع باستخدام الدالة (str())

إذا حاولت دمج الأرقام مع السلاسل النصية مباشرةً، فسيؤدي ذلك إلى خطأ. يجب تحويل الأرقام إلى نصوص باستخدام الدالة str(). في بايثون، يمكن أن تحتوي الأسطر البرمجية على أنواع بيانات مختلفة، مثل الأرقام والسلاسل النصية، ولكن عند محاولة دمج هذه الأنواع (مثل النصوص والأرقام)، يجب أن تكون بنفس النوع. في المثال أدناه، تم تحويل العدد 23 (عدد صحيح) إلى سلسلة نصية باستخدام str() ليصبح من الممكن دمجه مع السلسلة النصية.

بايثون لا يستطيع تحويل السلاسل الحرفية التي تحتوي على أحرف أو رموز غير رقمية إلى أعداد صحيحة عند وجود نوعين مختلفين من البيانات في السطر البرمجي الواحد بل تحويل الاعداد الى سلاسل حرفية.

Examples:

```
age = 23
message = "Happy " + str(age) + "rd Birthday!"
print(message)  # يطبع "Happy 23rd Birthday!"
```

بايثون يفهم الأرقام كأعداد صحيحة أو عشرية بشكل افتراضي، لكن إذا كانت مكتوبة داخل علامات اقتباس ("10")، فإنه يعاملها كسلاسل نصية، ويجب تحويلها إلى أعداد صحيحة باستخدام int() عند الحاجة لاستخدامها كأرقام.

```
num1 = "10"
num2 = "10"
result = int(num1) + int(num2)  # تحويل النصوص إلى أرقام
print(result)# يطبع "20"
```

اذا لم يتم استخدام int هنا فسيكون الناتج 1010 و ليس 20.

إذا تم إزالة علامات الاقتباس، فإن الأرقام ستُعتبر كأعداد صحيحة أو عشرية بشكل افتراضي في بايثون، ولا حاجة لتحويلها باستخدام int().

```
num1 = 10
num2 = 10
result = num1 + num2
print(result)  #  يطبع "20"
```

إذا كانت السلسلة تحتوي على حروف أو رموز غير رقمية، فإن محاولة تحويلها باستخدام int() ستؤدي إلى خطأ.

```
num_str = "10abc"
num = int(num_str)  # سيؤدي هذا إلى خطأ لأنه يحتوي على حروف
```

الخلاصة:

- إذا كان الهدف هو **دمج النصوص** مع الأرقام: استخدم str() لتحويل الأرقام إلى سلاسل حرفية.

- إذا كان الهدف هو **العمليات الحسابية** :استخدم int() لتحويل السلاسل الحرفية (التي تحتوي على أرقام ذات علامات اقتباس) إلى أعداد صحيحة.

التعليقات Comments

التعليقات تساعد في شرح الكود وتوثيقه، ولا يتم تنفيذها من قبل المفسر. يتم كتابة التعليقات باستخدام علامة (#).

Example:

```
# هذا تعليق لشرح الكود
print("Hello Python people!")  # يطبع رسالة ترحيب
```