Type Conversion in Python

Lecture 11
Understanding how data is converted from one type to another.

Agenda

1. What is Type Conversion?
2. Implicit Type Conversion
3. Explicit Type Conversion (Type Casting)
4. Built-in Casting Functions (‘int()‘, ‘float()‘, ‘str()‘)
5. Edge Cases and Conversion Errors
6. Practical Examples

What is Type Conversion?

The process of converting the value of one data type (integer, string, float, etc.) to another data type.
Also known as **Type Casting**.
Why do we need it?
- Receiving input from a user (which is always a string) and performing math on it.
- Combining different data types in output strings.
- Preventing data loss during mathematical operations.

Two Types of Conversion

Python handles type conversion in two ways:
1. Implicit Type Conversion (Coercion):
- Python automatically converts one data type to another.
- No user involvement is needed.
2. Explicit Type Conversion (Type Casting):
- The programmer manually converts the data type using built-in functions.
- Necessary when Python cannot automatically resolve type mismatches.

Implicit Type Conversion

When operating on mixed numeric types, Python automatically converts the smaller type to the larger/wider type to prevent data loss.

```
num_int = 123
num_float = 1.23

# Adding int and float
num_new = num_int + num_float

print("Type of num_new:", type(num_new))
# Output: <class 'float'>
print("Value:", num_new)
# Output: 124.23
```

Limitations of Implicit Conversion

Python cannot automatically convert entirely different data types, like strings and integers.

```
num_int = 123
num_str = "456"

# This will cause a TypeError
result = num_int + num_str
```

Error: ‘TypeError: unsupported operand type(s) for +: 'int' and 'str'‘
This is where Explicit Conversion is required.

Explicit Type Conversion (Casting)

We use built-in functions to force the conversion.
Common Functions:
- ‘int(a)‘: Converts ‘a‘ to an integer.
- ‘float(a)‘: Converts ‘a‘ to a float.
- ‘str(a)‘: Converts ‘a‘ to a string.
- ‘bool(a)‘: Converts ‘a‘ to a boolean.

Example: Casting String to Integer

Solving the TypeError from earlier:

```
num_int = 123
num_str = "456"

# Convert string to integer explicitly
num_str_converted = int(num_str)

result = num_int + num_str_converted
print("Sum:", result)
# Output: Sum: 579
```

Example: Casting Numbers to String

Useful for string concatenation.

```
age = 20

# TypeError: can only concatenate str (not "int") to str
# message = "I am " + age + " years old"

# Correct way using explicit conversion
message = "I am " + str(age) + " years old"
print(message)
```

(Note: Modern Python usually handles this with f-strings, which implicitly convert to string: ‘f”I am {age}”‘)

Loss of Data in Conversion

Explicit conversion can sometimes result in data loss.
Converting a float to an int **truncates** the decimal part.

```
pi = 3.14159
integer_pi = int(pi)

print(integer_pi)
# Output: 3
```

It does not round; it completely drops the decimal values.

Conversion Errors (ValueError)

Explicit conversion will fail if the data cannot be logically converted.

```
text = "Hello"
num = int(text)
```

Error: ‘ValueError: invalid literal for int() with base 10: ’Hello’‘
You can only cast strings to ints if the string contains only digits.
Similarly, ‘int(”3.14”)‘ will fail. You must cast to float first.

Summary

Implicit Conversion: Python handles it automatically (e.g., int + float = float).
Explicit Conversion: Programmer uses functions like ‘int()‘, ‘float()‘, ‘str()‘.
Explicit conversion is required when working with incompatible types (like string + int).
Be aware of potential data loss (float to int) and ValueErrors.