perplexity

# Notes on Data Types in Python

## Table of Contents



## 1. Introduction to Data Types

- **Definition:** In programming, a *data type* tells Python what kind of value a variable holds and what you can do with it.
- **Why Important?**: Understanding data types helps prevent errors and supports efficient code—like adding numbers vs. concatenating strings.

**Key Python Data Types:**

- Numbers
- Boolean
- String
- Sequence (List, Tuple, Range)

- Mapping (Dict)
- Set
- NoneType

## 2. Numbers

Numbers in Python are used to store numeric values and come in three main forms:

### 2.1 Integer (`int`)

- Whole numbers, positive or negative, without decimals

```
age = 25         # int
temperature = -3 # int
big_number = 12345678901234567890 # int (Python int has unlimited precision)
print(type(age))  # Output: <class 'int'>
```

### 2.2 Floating Point (`float`)

- Numbers with decimal points

```
height = 5.9   # float
price = -12.75 # float
division = 10 / 3
print(division)        # Output: 3.3333333333333335
print(type(division))  # Output: <class 'float'>
```

### 2.3 Complex Number (`complex`)

- Real and imaginary components

```
z = 2 + 3j      # complex number, real part is 2, imaginary part is 3
print(z.real)   # Output: 2.0
print(z.imag)   # Output: 3.0
print(type(z))  # Output: <class 'complex'>
```

## 3. Boolean (`bool`)

- Represents one of two values: `True` or `False`

- Used in conditions, comparisons, and logic

```
is_active = True
is_closed = False
print(type(is_active))          # Output: <class 'bool'>

# Boolean from comparison
print(5 > 3)    # Output: True
print(2 == 10)  # Output: False
```

### 4. String (`str`)

- A sequence of characters (letters, numbers, symbols)
- Defined using single, double, or triple quotes

```
name = "Python"
greeting = 'Hello, world!'
multi_line = '''This is
a multi-line
string.'''

print(name[0])     # First character: Output 'P'
print(name[-1])    # Last character: Output 'n'
print(len(name))   # Output: 6
print(name.upper())# Output: 'PYTHON'
```

**String Operations:**

```
first = "Hello"
second = "World"
combined = first + " " + second
print(combined)             # Output: 'Hello World'
print(combined[6:])         # Output: 'World' (slicing)
```

### 5. Sequence Data Types

### 5.1 List (`list`)

- Ordered, mutable collection. Can contain elements of any data type.

```
numbers = [1, 2, 3, 4]
mixed = [1, "apple", 3.14, True]

print(numbers[2])        # Output: 3
numbers[0] = 10          # List is mutable, you can change values
print(numbers)           # Output: [10, 2, 3, 4]

# Adding and removing
numbers.append(5)        # Add to end
print(numbers)           # Output: [10, 2, 3, 4, 5]
numbers.remove(3)        # Remove value 3
print(numbers)           # Output: [10, 2, 4, 5]
```

### 5.2 Tuple (`tuple`)

- Ordered, immutable collection. Once created, cannot be changed.

```
coordinates = (12.4, 77.6)
one_item = (5,)      # Note the comma for single element tuple
print(type(coordinates))      # Output: <class 'tuple'>

# Trying to change value will give error
# coordinates[0] = 100  # Error: 'tuple' object does not support item assignment
```

### 5.3 Range (`range`)

- Represents a sequence of numbers; commonly used in loops.

```
numbers = range(0, 5)     # 0, 1, 2, 3, 4 (5 not included)
for i in numbers:
    print(i)              # Prints from 0 to 4

# Convert to list for viewing
print(list(numbers))      # Output: [0, 1, 2, 3, 4]
```

### 6. Mapping Data Type

### Dictionary (`dict`)

- Unordered, changeable (mutable), key-value pairs

```
student = {
    "name": "Sam",
    "age": 21,
    "is_active": True
}

print(student["name"])    # Output: Sam
student["age"] = 22       # Update value
print(student)            # Output: {'name': 'Sam', 'age': 22, 'is_active': True}

# Adding a new key-value pair
student["grade"] = 'A'
print(student)

# Iterating over dictionary
for key, value in student.items():
    print(key, ":", value)
```

## 7. Set Data Types

### 7.1 Set (`set`)

- Unordered, mutable, no duplicates

```
fruits = {"apple", "banana", "cherry", "apple"} # 'apple' appears only once
print(fruits)           # Output: {'apple', 'cherry', 'banana'}

fruits.add("mango")
fruits.remove("banana")
print(fruits)           # Output: {'apple', 'mango', 'cherry'}
```

### 7.2 Frozenset (`frozenset`)

- Like set, but immutable

```
fset = frozenset([1, 2, 3, 3])
print(fset)              # Output: frozenset({1, 2, 3})
```

```
# Not allowed:
# fset.add(4)            # Error: 'frozenset' object has no attribute 'add'
```

### 8. None Type

- `None` is a special type representing the absence of a value.

```
result = None
print(result)              # Output: None
print(type(result))        # Output: <class 'NoneType'>
```

### 9. Type Conversion (Casting)

- Converting between data types using built-in functions: `int()`, `float()`, `str()`, `list()`, etc.

```
x = 5
y = float(x)          # int to float: y = 5.0
z = str(x)            # int to str: z = '5'

print(type(x), type(y), type(z))  # Output: <class 'int'> <class 'float'> <class 'str'>

# Example: string to int
num_str = "123"
num = int(num_str)
print(num + 10)   # Output: 133

# List to set
nums = [1,2,3,3]
unique_nums = set(nums)
print(unique_nums)    # Output: {1, 2, 3}
```

### 10. Checking & Identifying Data Types

- Use `type()` function to check the data type of a value
- Use `isinstance()` to check if a variable is of a certain type

```
print(type(4.5))             # Output: <class 'float'>
print(isinstance(4.5, float)) # Output: True
```

```
# Useful in conditional logic
a = "hello"
if isinstance(a, str):
    print("a is a string")
```

## Summary Table of Basic Python Data Types

| Data Type | Example Value | Description |
|---|---|---|
| int | `10`, `-4`, `1000` | Integer numbers |
| float | `3.14`, `-0.01` | Decimal numbers |
| complex | `2+3j` | Complex numbers |
| bool | `True`, `False` | Boolean values |
| str | `'hello'`, `"yes"` | Strings (text) |
| list | `` | Mutable sequence |
| tuple | `(4,5)` | Immutable sequence |
| dict | `{'a':1, 'b':2}` | Key-value pairs |
| set | `{'a','b'}` | No duplicates, unordered |
| frozenset | `frozenset()` | Immutable set |
| NoneType | `None` | No value/empty |
| range | `range(5)` | Sequence of numbers (0-4) |

## Quick Data Type Practice

Test yourself—guess the type **before** running the code!

```
a = [1, 2, 3]     # ?
b = (1, 2, 3)     # ?
c = {'x': 100}    # ?
d = {1, 2, 3}     # ?
e = None          # ?
```

```
print(type(a), type(b), type(c), type(d), type(e))
```

### Tips to Remember

- Python has dynamic typing: variables can change their type at runtime.
- Use `type()` and `isinstance()` often to debug or check data types.
- Mixing data types in operations may cause errors (e.g., cannot add a number to a string).
- Practice creating, modifying, and checking all these types to build strong coding intuition!

# End of Notes

**If you master these Python data types, you'll lay the strongest foundation for becoming a Python pro! Practice each type, experiment, and always explore with print statements. Happy coding!** 🚀