Variables and Basic Data Types

Variables in Python

Creating variables in Python

Syntax:

<***variable-name***> **=** <***value or literal***>

Assignment operator

E.g.:

```
x = 123
```

Code Demo - Variable Creation

Rules for naming Variables in Python

1. A variable name can only contain alphanumeric characters and underscores (A-z, 0-9, and _).
   E.g. **`foo, foo123, bar_1`** ✅ **`b@r$, age-sum`** ❌

1. It must start with a letter or an underscore, but not with a number.
   E.g. **`foo, _foo`** ✅ **`1bars, 9foo`** ❌

1. Variable names are case-sensitive.
   E.g. **`foo, Foo`** and **`FOO`** are three different variables.

1. Reserved keywords shouldn't be used as variable names.
   E.g. **`def, pass, continue, break`** ❌

Python Keywords

```
False                 await                 else
        import                pass

None                  break                 except                in
                              raise

True                  class                 finally               is
                              return

and                           continue              for
                      lambda                try

as                            def                                 from
                              nonlocal              while

assert                del                           global
        not                           with

async                 elif                          if
```

Basic Python Data Types

Basic Data Types

1. **Numeric** data type
2. **String** data type
3. **Boolean** data type
4. **None** data type

Basic Data Types

1. **Numeric** data type
2. **String** data type
3. **Boolean** data type
4. **None** data type

Numeric data type

- Integer, Floating-point, Complex numbers
  - `123` (integer)
  - `3.1416` (float)
  - `2 + 3j` (complex)
- **Arithmetic Operators**
  - '+' - Addition
  - '–' - Subtraction
  - '*' - Multiplication
  - '/' - Division
  - '%' - Modulus
  - '**' - Exponentiation

Code Demo - Numeric Data Type

Basic Data Types

1. **Numeric** data type
2. **String** data type
3. **Boolean** data type
4. **None** data type

String data type

- Strings represents texts.
- **string literal** - a collection of characters, enclosed in single quotes ('...') or double quotes ("...")
- E.g.
  - `'Hello, World!'`
  - `"Hello, World!"`

string literal

Code Demo - String Data Type

Basic Data Types

1. **Numeric** data type
2. **String** data type
3. **Boolean** data type
4. **None** data type

Boolean data type

- Boolean literal is either **True** or **False**
- Logical Operators:
  - `and` - Boolean AND Operation
  - `or` - Boolean OR Operation
  - `not` - Boolean Not Operation
- Comparison Operators output boolean values.
  - `(==, is, !=, <, >, <=, >=)`

Code Demo - Boolean Data Type

Basic Data Types

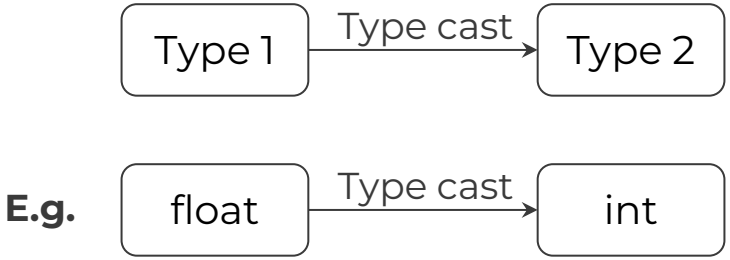
1. **Numeric** data type
2. **String** data type
3. **Boolean** data type
4. **None** data type

None data type

- signifies the absence of a value in many situations.
- accessed through the built-in keyword **None**

Type-casting in Python

Type-casting in Python



1. **Implicit** Cast: done by the interpreter
2. **Explicit** Cast: done using predefined functions by the programmer

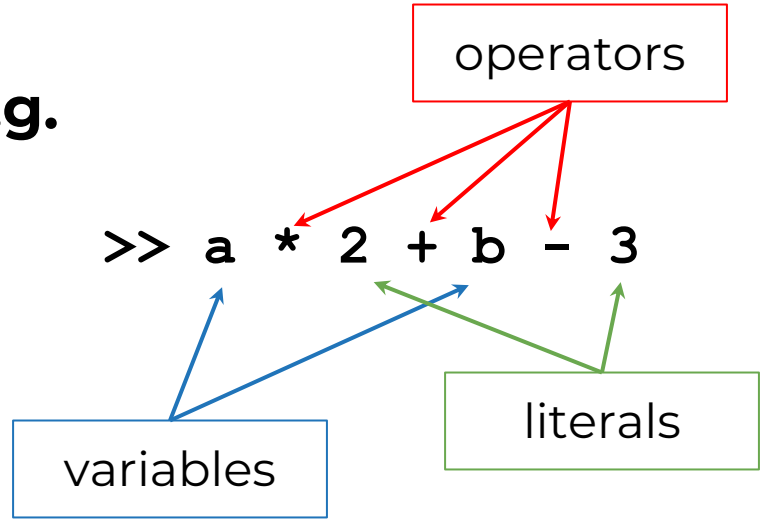
Code Demo - Type casting

Expression and Statement

An **expression** is a collection of variables, literals and operators.

E.g.

```
>> a * 2 + b - 3
```



A **statement** is a unit of code that has an effect, like displaying a value or creating a variable.

E.g.

```
>> x = 2

>> print(x)

>> y = x ** 2
```

3 statements

Operators

Operators

Arithmetic Operators

Operator	Description
`+`	Addition
`-`	Subtraction
`*`	Multiplication
`/`	Division
`//`	Integer Division
`**`	Exponentiation
`%`	Modulus

Logical Operators

Operator	Description
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

Comparison Operators

Operator	Description
`==`	Equal
`!=`	Not equal
`<`	Less than
`>`	Greater than
`<=`	Less than or equal
`>=`	Greater than or equal

Operator Precedence

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x, -x,`	Unary plus, Unary minus
`*, /, //, %`	Multiplication, Division, Floor Division, Remainder
`+, -`	Addition, Subtraction
`==, !=, >, >=, <, <=`	Comparison operators
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

Precedence

```
>> 2 + 3 / 1.5
```

Operator Precedence

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x, -x,`	Unary plus, Unary minus
`*, /, //, %`	Multiplication, Division, Floor Division, Remainder
`+, -`	Addition, Subtraction
`==, !=, >, >=, <, <=`	Comparison operators
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

Precedence

Operators

```
>> 2 + 3 / 1.5
   = 2 + 2.0
   = 4.0
```

Operator Precedence

Operator	Description
`()`	Parentheses
`**`	Exponentiation
`+x, -x,`	Unary plus, Unary minus
`*, /, //, %`	Multiplication, Division, Floor Division, Remainder
`+, -`	Addition, Subtraction
`==, !=, >, >=, <, <=`	Comparison operators
`not`	Logical NOT
`and`	Logical AND
`or`	Logical OR

Precedence

Operators

```
>> 2 + 3 / 1.5
   = 2 + 2.0
   = 4.0
```

Equivalent Expressions

```
>> 2 + (3 / 1.5)
```

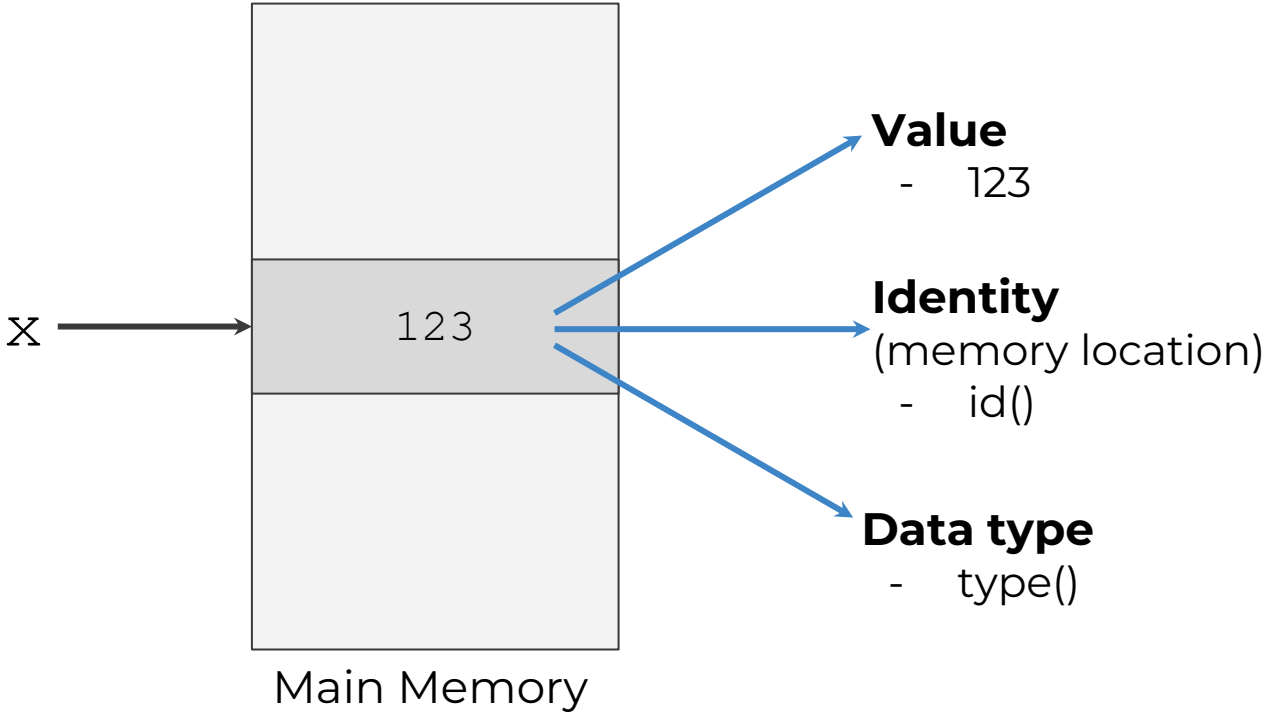
```
>> (2 + 3) / 1.5
```

Adds 2 and 3 before division

Variable Binding in Python

A variable references an object in the main memory and the object has a **value**, **identity** and a **type**.

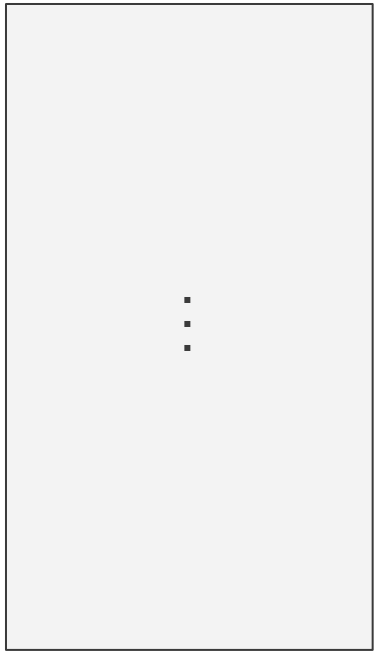
```
>> x = 123
```



Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet

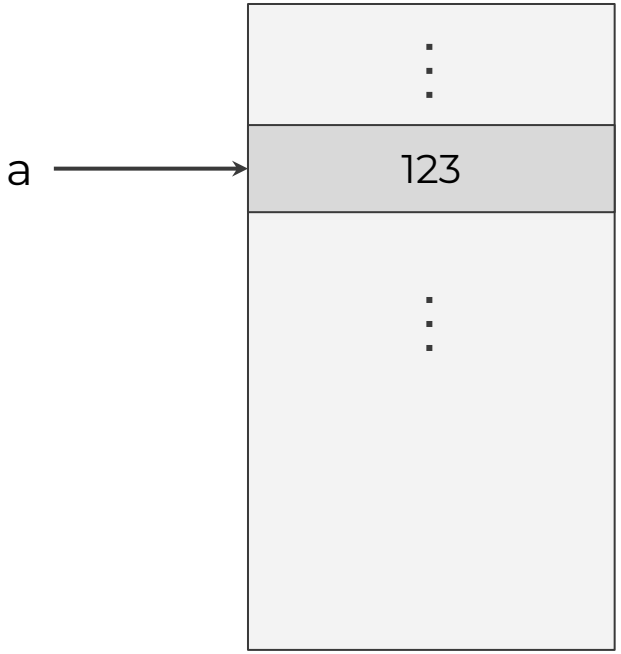


Main Memory

Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet

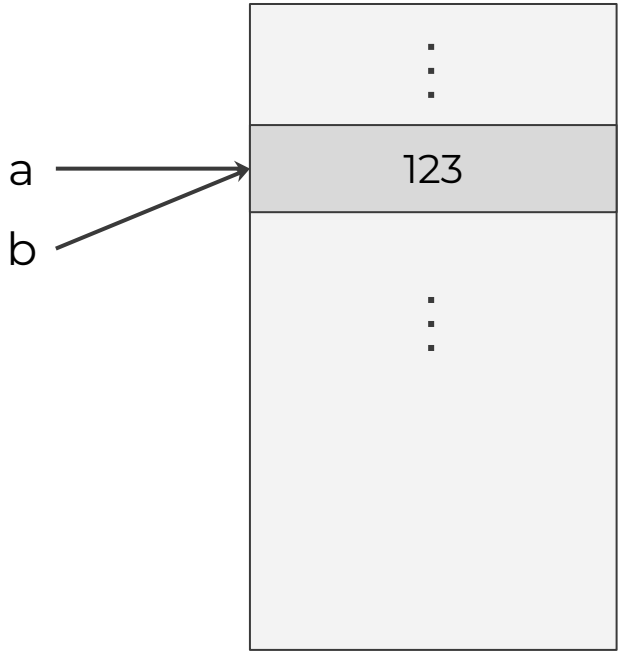


Main Memory

Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet

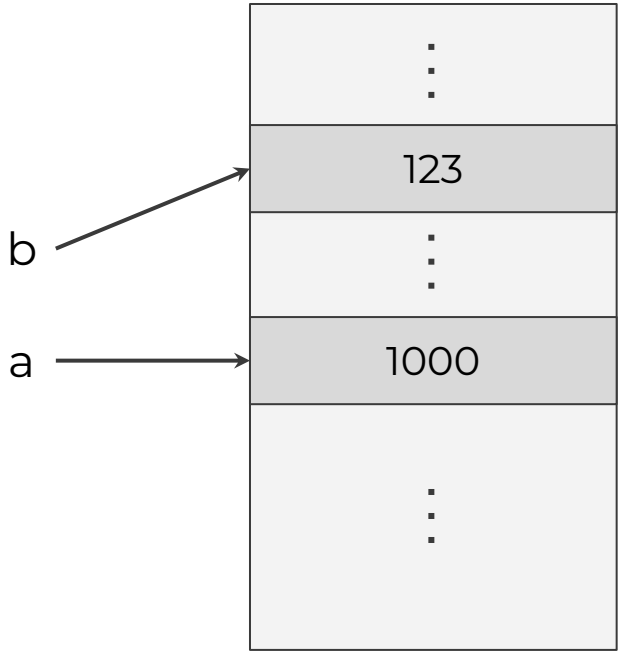


Main Memory

Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet

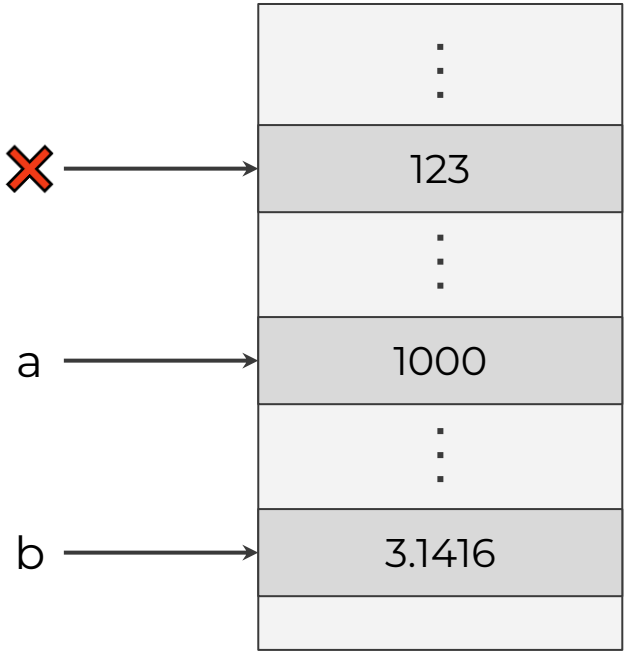


Main Memory

Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet

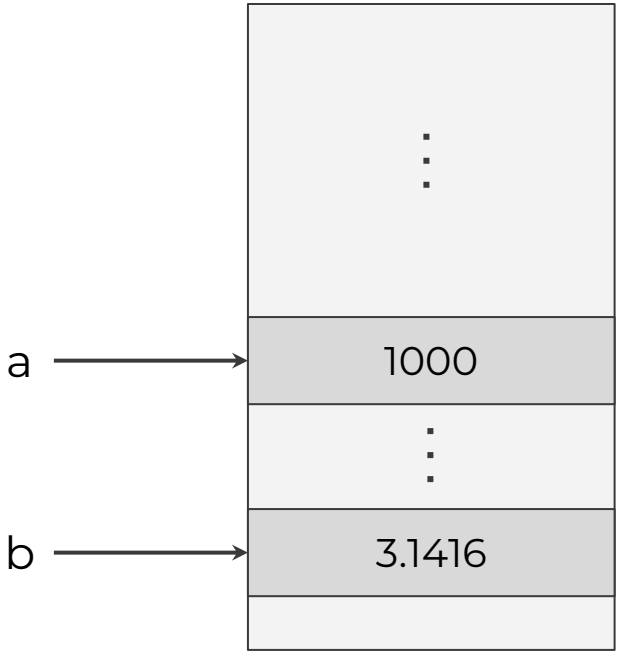


Main Memory

Variable Binding in Python

```
a = 123
b = a
a = 1000
b = 3.1416
```

Code Snippet



Main Memory

Code Demo - Variable Binding

Key Takeaways

1. Variable naming syntax: **<variable-name> = <value>**
2. Different data types: **int, float, complex, string, bool, None.**
3. Operators- **Arithmetic, Comparison** and **Logical**.
4. A variable references an object in the main memory and the object has a value, identity and a type.