Introduction to Computer Programming
Lecture 2.1:

Variables & Types

Hemma Philamore

Department of Engineering Mathematics

Variables

Variable name

StudentNumber = 12345

Height = 180.5

Value

Age = 23

Assigning

Case matters!

“Age” is not the same as “age” !!!

Variables

```
Result = Height * (3 + Age/2)
```

Result is overwritten!

everything on the right is evaluated and assigned to the left

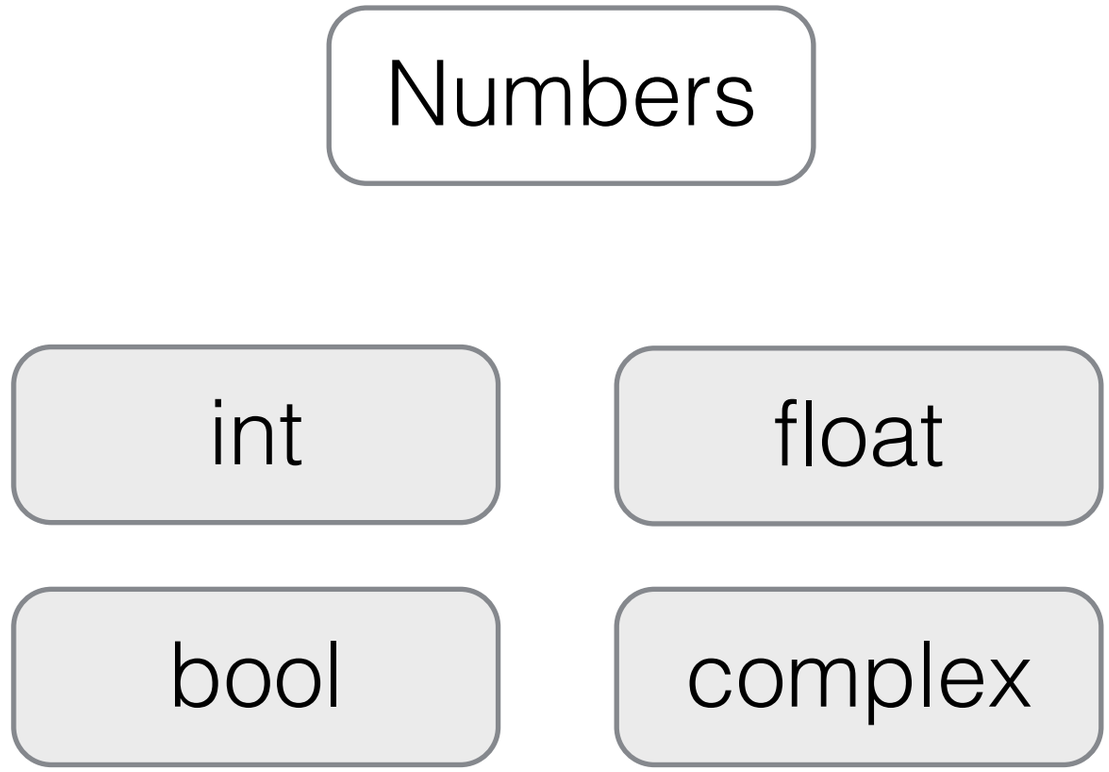
```
a = 3
a = a +1
a += 1
```

variable a is assigned to value 3

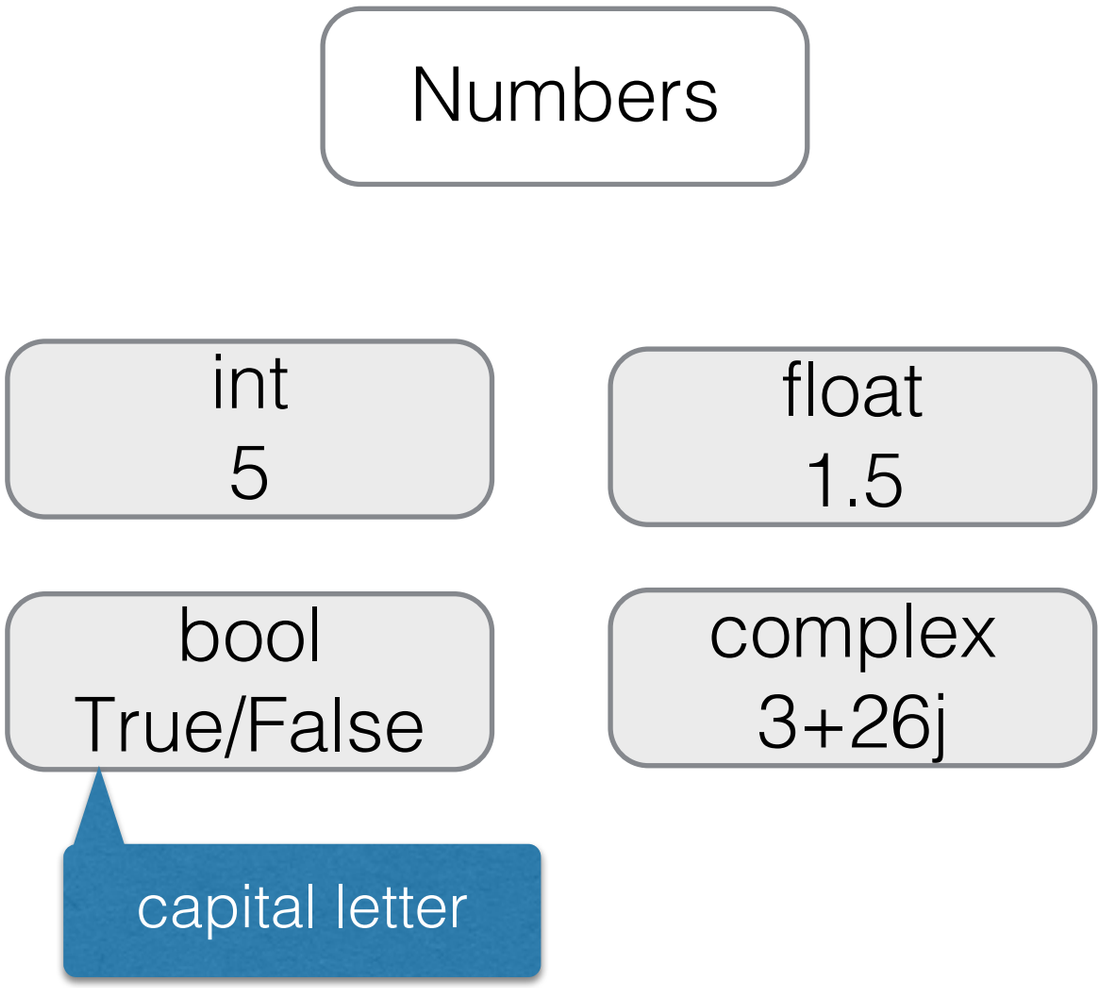
a + 1 is replaced by 3 +1 => 4

a is assigned to new value of 4

Numbers in Python



Numbers in Python



int: Integer number.

float: Number with a decimal point.

complex: Number with a ’j’.

bool: True or False.

There are also ***bin, oct and hex***
(binary, octal and hexadecimal representation)

Types of variables

How does Python know what type you want?

```
>>> a = 2
>>> type(a)
<class 'int'>
```

Decimal point tells Python it's a ***float***

```
>>> b = 2.0
>>> type(b)
<class 'float'>
```

Quotation marks tell Python it's a string

```
>>> c = "2.0"
>>> type(c)
<class 'str'>
```

capital letter

```
>>> d = True
>>> type(d)
<class 'bool'>
```

orange color, IDLE recognises it as a keyword

```
>>> type(3<4)
<class 'bool'>
```

```
>>> f = 2+3j
>>> type(f)
<class 'complex'>
```

comparison results in boolean

Mixing types doesn't always work

```
>>> a = 3
>>> type( a )
<class “int“>

>>> b = 3.0
>>> type( b )
<class “float“>

>>> d = “3“
>>> type( d )
<class “str“>
```

```
>>> e = b + a
>>> e = d + a
Traceback (most recent call last):
File “<input>“, line 1, in <module>
TypeError: must be str, not int
```

Context is important

Operators behave differently depending on the types of the operands

```
>>> 2+3
5
```

```
>>> “2“ + “3“
23
```

```
>>> “2“ - “3“
Traceback (most recent call last):
File “<pyshell#34>“, line 1, in <module>
      “2“-“3“
TypeError: unsupported operand type(s) for -: “str“ and “str“
```

Some operators won’t work if they are called on the wrong type.

e.g. minus (-) doesn’t make sense for strings

Checking and Casting Types

isinstance() and ***type()*** allow you to check types.

```
>>> type(1)
<class 'int'>
>>> isinstance(2.5,float)
True
```

int, float, complex, bool, str

You can sometimes **explicitly cast** types to change from one type to an other (**str, float, bool**)

```
>>> A = True
>>> B = int(A)
>>> B
1
>>> type(B)
<class 'int'>
```

```
>>> w = "Hello"
>>> a = 1
>>> r = w + str(a)
>>> r
'Hello1'
```

Number operators

Math operators

+	Addition
−	Subtraction
*	Multiplication
/	Division
//	Floor division (round down to the next integer)
%	Modulo (remainder)
**	Exponent

% modulo

```
>>> 10 % 3
1
>>> 10 % 5
0
>>> 1+4-2*4/2
1.0
```

multiplication

```
>>> 3*4
12
```

exponent

```
>>> 2**4
16
```

// floor division

```
>>> 8//5
1
```

additional functionality

```
>>> import math
>>> math.sin(1)
0.8414709848078965
```

Number operators

Boolean operators

==	Equality
! =	Inequality
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to

```
>>> 1 > 2
False
>>> 3 <= 5
True
>>> 1 == 1
True
>>> 1 != 2
True
```

```
>>> True and False
False
>>> True or False
True
>>> not False
True
>>> not (1==1)
False
```

Number operators

Logical operators: compare the outcome of two conditionals

and : both are true
or : either are true

not: negates the outcome of a conditional

```
>>> (1 > 2 and 3 < 4)
False

>>> (1 > 2 or 3 < 4)
True

>>> not(1 > 2 or 3 < 4)
False

>>> not(1 > 2) and 3 < 4
True
```

Number operators

Math operators

% modulo

// divide and discard remainder

```
>>> 10 % 3
1
>>> 10 % 5
0
>>> 1+4-2*4/2
1.0
>>> 8//5
1
```

```
>>> 3*4
12
>>> 2**4
16
>>> import math
>>> math.sin(1)
0.8414709848078965
```

additional functionality

Boolean operators

Note that it's "== "and not "=" !

```
>>> 1 > 2
False
>>> 3 <= 5
True
>>> 1 == 1
True
>>> 1 != 2
True
```

```
>>> True and False
False
>>> True or False
True
>>> not False
True
>>> not (1==1)
False
```

Time-telling program

Based on the current time of day, the program answers two questions:

> **Is it lunchtime?**

> `True`

if it is lunch time.

> **Is it time for work?**

> `True`

if it is `not`:

- before work (`time < work_starts`)
- after work (`time > work_ends`)
- lunchtime (the previous question assigns the value `True` or `False` to variable `lunchtime`).

```python
# Time-telling program

time = 13.05           # current time

work_starts = 8.00     # time work starts
work_ends =  17.00     # time work ends

lunch_starts = 13.00   # time lunch starts
lunch_ends =   14.00   # time lunch ends

# lunchtime if the time is between the start and end of lunchtime
lunchtime = time >= lunch_starts and time < lunch_ends

# work_time if the time is not...
work_time = not (   time < work_starts      # ... before work
                 or time > work_ends        # ... or after work
                 or lunchtime)              # ... or lunchtime


print("Is it work time?", work_time)
print("Is it lunchtime?", lunchtime)
```

```
Is it work time? False
Is it lunchtime? True
```

Summary

Data Types

- We can *assign* values to variables.
- Every variable has a type (`int` , `float` , `string` ....).
- A type is automatically assigned when a variable is created.
- Python's `type()` function can be used to determine the type of a variable.
- The data type of a variable can be converted by casting (`int()` , `float()` ....)

Summary

Math Operators

We can perform simple *arithmetic operations* in Python (+, -, ×, ÷.....)

Boolean Operators

- *Comparison operators* (==, !=, <, >....) compare two **variables**.
- The outcome of a comparison is a *Boolean* (True or False) value.
- *Logical operators* (`and` , `or`) compares the outcomes of two **comparison operations**.
- The outcome of a logical operation is a *Boolean* (True or False) value.
- The logical `not` operator returns the inverse Boolean value of a comparison.