Introduction to python

Rossano Venturini
rossano.venturini@unipi.it
19 hours lecture
20 hours laboratory
Python (programming language)

From Wikipedia, the free encyclopedia

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.

The language provides constructs intended to enable writing clear programs on both a small and large scale. Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.
Python (programming language)

From Wikipedia, the free encyclopedia

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.[23][24] The language provides constructs intended to enable writing clear programs on both a small and large scale.[25]

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.[26]

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software[27] and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.
Python (programming language)

From Wikipedia, the free encyclopedia

Python is a widely used high-level programming language for general-purpose programming, created by Guido van Rossum and first released in 1991. An interpreted language, Python has a design philosophy that emphasizes code readability (notably using whitespace indentation to delimit code blocks rather than curly brackets or keywords), and a syntax that allows programmers to express concepts in fewer lines of code than might be used in languages such as C++ or Java.[23][24] The language provides constructs intended to enable writing clear programs on both a small and large scale.[25]

Python features a dynamic type system and automatic memory management and supports multiple programming paradigms, including object-oriented, imperative, functional programming, and procedural styles. It has a large and comprehensive standard library.[26]

Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems. CPython, the reference implementation of Python, is open source software[27] and has a community-based development model, as do nearly all of its variant implementations. CPython is managed by the non-profit Python Software Foundation.
Python (programming language)

From Wikipedia, the free encyclopedia

Python is a widely used high-level programming language with dynamic semantics and strong typedness. Developed by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability (notably using significant whitespace) and its syntax allows programmers to express concepts in fewer lines of code than would be possible in languages such as C++ or Java.[23][24] The language supports multiple programming paradigms, including imperative, object-oriented, and functional programming, and it has a comprehensive standard library.

Python interpreters and implementations based on the Python C runtime, such as CPython, the reference implementation, and PyPy, have become widely used for general-purpose programming and scripting. Python is used in many areas including web (and especially web application) development, network programming, system administration, and embedded system development.

<table>
<thead>
<tr>
<th>I DUNNO... DYNAMIC TYPING? WHITESPACE?</th>
</tr>
</thead>
<tbody>
<tr>
<td>I JUST TYPED import antigravity</td>
</tr>
<tr>
<td>THAT'S IT?</td>
</tr>
<tr>
<td>... I ALSO SAMPLED EVERYTHING IN THE MEDICINE CABINET FOR COMPARISON.</td>
</tr>
<tr>
<td>BUT I THINK THIS IS THE PYTHON.</td>
</tr>
</tbody>
</table>

| I LEARNED IT LAST NIGHT! EVERYTHING IS SO SIMPLE! HELLO WORLD IS JUST print "Hello, world!" |
| COME JOIN US! PROGRAMMING IS FUN AGAIN! IT'S A WHOLE NEW WORLD UP HERE! BUT HOW ARE YOU FLYING? |
| YOU'RE FLYING! HOW? |
| OK |
| PYTHON! |

<table>
<thead>
<tr>
<th>Paradigm</th>
<th>Design by</th>
<th>Development Foundation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>First approach</td>
</tr>
</tbody>
</table>
• Python programs are text files

• By convention, the name of those files have the extension .py

• You will find a lot of resources on the web, including tutorial

• e.g., http://learnpython.org/

The Coder’s Apprentice

*Learning Programming with Python 3*
di Pieter Spronck
www.spronck.net/pythonbook
Tools you will need

- Python 3 on your virtual machine
- Jupyter (run with jupyter notebook) + Web browser
- Spyder (Development environment)
- Editor
- Debugger
# coding: utf-8

Spyder Editor

This is a temporary script file.

print("Hello World")
Editor

Console (or shell)
Let’s start cooking!
Let’s start cooking!

**Ingredienti per la pasta frolla**
- Farina 400 g
- Uova 4 tuorli
- Zucchero al velo 150 gr
- Burro 200 g
- Sale 1 pizzico
- Vaniglia 1 bacca

**Ingredienti per la crema pasticcera**
- Latte fresco 1 litro
- Uova 8 tuorli
- Farina 80 g
- Zucchero 250 g
- Limoni la scorza di 1/2
- Vaniglia 1 bacca

**Ingredienti per la ricopertura**
- Pinoli 120 g
- Zucchero a velo vanigliato q.b.
Let’s start cooking!

**Ingredienti per la pasta frolla**
- Farina 400 g
- Uova 4 tuorli
- Burro 200 g
- Zucchero al velo 150 gr
- Sale 1 pizzico
- Vaniglia 1 bacca

**INGREDIENTI PER LA CREMA PASTICCERA**
- Latte fresco 1 litro
- Uova 8 tuorli
- Limoni la scorza di 1/2
- Farina 80 g
- Zucchero 250 g
- Vaniglia 1 bacca

**INGREDIENTI PER LA RICOPERTURA**
- Pinoli 120 g
- Zucchero a velo vanigliato q.b.
Let’s start cooking!

**Ingredienti per la pasta frolla**

- Farina 400 g
- Zucchero al velo 150 gr
- Sale 1 pizzico
- Uova 4 tuorli

**Ingredienti per la crema**

- Latte fresco 1 litro
- Limoni la scorza di 1/2

**Ingredienti per la ricetta**

- Pinoli 120 g

Preparate la pasta frolla seguendo il procedimento indicato [clicca qui](#), ma attenendovi alle dosi di questa ricetta. Una volta pronta la pasta frolla, datele la forma di un panetto basso, avvolgetela nella pellicola trasparente e ponetela in frigorifero per almeno un'ora.

Nel frattempo preparate la crema pasticcera seguendo il procedimento indicato [clicca qui](#) sempre attenendovi alle dosi di questa ricetta.

Una volta pronta la crema pasticcera, lasciatela raffreddare in frigorifero in una teglia bassa coperta da una pellicola che deve rimanere a contatto con la crema, di modo che non si formi una pellicina, estraete dal frigorifero la pasta frolla, prendete dall'intero impasto 2/3 della stessa (1) e stendetela in un disco (2) che possa foderare (coprendo anche i lati) una tortiera (3) del diametro di circa 24-26 cm.

Imburrate e infarinata la tortiera, quindi avvolgete il disco di pasta frolla sul matterello e srotolate sulle tortiera, coprendo così fondo e lati.
Everybody starts like this: Hello, World!
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```
Everybody starts like this: Hello, World!

In a text editor:

```
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
```
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$ 
```
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$ 
```
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$ 
```
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$ 
```

what if we want to print more than one sentence?
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$
```

what if we want to print more than one sentence?

```python
print("Hello, World!")
print("How are you?")
```
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!")
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$
```

what if we want to print more than one sentence?

```python
print("Hello, World!")
print("How are you?")
```

The program will be executed one statement after the other.
Everybody starts like this: Hello, World!

In a text editor:

```python
print("Hello, World!"
```

Save and run from the terminal the command `python hw.py`

```
$ python hw.py
Hello, World!
$
```

what if we want to print more than one statement?

```
print("Hello, World!")
print("How are you?"
```

The program will be executed one statement after the other
The \texttt{print} function

- The syntax is quite easy

\texttt{print(“whatever you want to display”)}
The print function

• The syntax is quite easy

print("whatever you want to display")
The print function

• You multiple things with one print() function

```python
print("I", "am", "having", "fun", ")
```
The **print** function

• You multiple things with one `print()` function

```
print(“I”, “am”, “having”, “fun”, “!”)

print( “I”, “am”, “having”, “fun”, “!” )

print ( “I”, “am”, “having”, “fun”, “!” )
```

Spaces superfluous
The `print` function

- You multiple things with one `print()` function

```
print(2, "+", 2, "is", 4)
```

```
print(2, "+", 2, "is", "4")
```
Python can even count!
Python can even count!

```python
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```
Python can even count!

```python
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```
Python can even count!

```python
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```
Python can even count!

```python
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```
Python can even count!

```python
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```

$ python example.py
Python can even count!

```python
print(“3 + 5 =”, 3 + 5)
print (“3 * 5 =”, 3 * 5)
print (“3 - 5 =”, 10 - 5)
```

```
$ python example.py
3+5 = 8
3*5 = 15
3-5 = 5
```
Python can even count!

```
print("3 + 5 =", 3 + 5)
print ("3 * 5 =", 3 * 5)
print ("3 - 5 =", 10 - 5)
```

```
$ python example.py
3+5 = 8
3*5 = 15
3-5 = 5
```

### 2.3 Espressioni

Ecco qui un’altro programma:

```
print "2 + 2 is", 2+2
print "3 * 4 is", 3 * 4
print 100 - 1, " = 100 - 1"
print "(33 + 2) / 5 + 11.5 = ",(33 + 2) / 5 + 11.5
```

E qui l’output che questo programma produce:

```
2 + 2 is 4
3 * 4 is 12
99 = 100 - 1
(33 + 2) / 5 + 11.5 = 18.5
```

Come puoi vedere Python può trasformare il vostro costosissimo computer in una normale calcolatrice :-)

Python ha sei operatori basilari:

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Simbolo</th>
<th>Esempio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevamento a potenza</td>
<td>**</td>
<td>5 ** 2 == 25</td>
</tr>
<tr>
<td>Moltiplicazione</td>
<td>*</td>
<td>2 * 3 == 6</td>
</tr>
<tr>
<td>Divisione</td>
<td>/</td>
<td>14 / 3 == 4</td>
</tr>
<tr>
<td>Resto</td>
<td>%</td>
<td>14 % 3 == 2</td>
</tr>
<tr>
<td>Addizione</td>
<td>+</td>
<td>1 + 2 == 3</td>
</tr>
<tr>
<td>Sottrazione</td>
<td>-</td>
<td>4 - 3 == 1</td>
</tr>
</tbody>
</table>
Python can even count!

- print("3 + 5 =", 3 + 5)
- print ("3 * 5 =", 3 * 5)
- print ("3 - 5 =", 10 - 5)

3 Statements
1 statement per line (standard convention in Python)
If you have more than one statement on the same line, you need ; after each one

$ python conti.py
3+5 = 8
3*5 = 15
3-5 = 5

---

<table>
<thead>
<tr>
<th>Operatore</th>
<th>Simbolo</th>
<th>Esempio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Elevamento a potenza</td>
<td>**</td>
<td>5 ** 2 == 25</td>
</tr>
<tr>
<td>Moltiplicazione</td>
<td>*</td>
<td>2 * 3 == 6</td>
</tr>
<tr>
<td>Divisione</td>
<td>/</td>
<td>14 / 3 == 4</td>
</tr>
<tr>
<td>Resto</td>
<td>%</td>
<td>14 % 3 == 2</td>
</tr>
<tr>
<td>Addizione</td>
<td>+</td>
<td>1 + 2 == 3</td>
</tr>
<tr>
<td>Sottrazione</td>
<td>-</td>
<td>4 - 3 == 1</td>
</tr>
</tbody>
</table>
Exercises

1. Write and run a program that displays your first and last name

2. Write and run a program that uses all math operations (pay attention to precedences)
2

Python: let’s get more serious

Rossano Venturini
rossano.venturini@unipi.it
Values and types

*Values* are the main objects handled by programmers
Values and types

Values are the main objects handled by programmers

42  “Hello, World!”  4.2

These values have different types
Values and types

Values are the main objects handled by programmers

These values have different types

- Integer
- String
- Float
Values and variables

Values are the main objects handled by programmers

42
“Hello, World!”
4.2

Integer String Float

These values have different types

A variable is a name that refers to a value
Values and variables

*Values* are the main objects handled by programmers.

These values have different *types*.

A *variable* is a name that refers a value.
Values and variables

*Values* are the main objects handled by programmers

These values have different *types*

A *variable* is a name that refers a value

The *assignment* creates new variables and assigns them values
Values and variables

Values are the main objects handled by programmers

42
“Hello, World!”
4.2

Integer  String  Float

These values have different types
A variable is a name that refers a value

The assignment creates new variables and assigns them values

```
message = “Hi”
print(message)
message = “Am I having fun?”
print(message)
```
Values are the main objects handled by programmers

These values have different types

A variable is a name that refers a value

The assignment creates new variables and assigns them values

```python
message = "Hi"
print(message)
message = "Am I having fun?"
print(message)
```
Values and variables

Values are the main objects handled by programmers.

These values have different types:
- Integer
- String
- Float

A variable is a name that refers to a value.

The assignment creates new variables and assigns them values.

Example:

```python
message = “Hi”
print(message)
message = “Am I having fun?”
print(message)
```
Values

Values are the main objects handled by programmers

42  “Hello, World!”  4.2

Integer  String  Float

These values have different types

A variable is a name that refers to a value

The assignment creates new variables and assigns them values

message = “Hi”
print(message)
message = “Am I having fun?”
print(message)

Displays the value of message
Variables can be manipulated using expressions

```python
x=5
print( x )
x = 7 * 9 + 13 # overwrite the previous value of x
print( x )
x = "A nod's as good as a wink to a blind bat."
print( x )
x = int( 15 / 4 ) - 27
print( x )
```
Variables can be manipulated using expressions

\[
x = 5
\]

```
x=5
print(x)
```

```
x = 7 * 9 + 13  # overwrite the previous value of x
print(x)
```

```
x = "A nod's as good as a wink to a blind bat."
print(x)
```

```
x = int(15 / 4) - 27
print(x)
```
Variables can be manipulated using expressions

```
x=5
print(x)  # 5
x = 7 * 9 + 13  # overwrite the previous value of x
print(x)  # 76
x = "A nod's as good as a wink to a blind bat."
print(x)
```

```
x = int( 15 / 4 ) - 27
print(x)
```
Variables can be manipulated using expressions

\[
x=5
\]

\[
\text{print}( \ x \ ) \quad 5
\]

\[
x = 7 \times 9 + 13 \quad \# \ \text{overwrite the previous value of } x
\]

\[
\text{print}( \ x \ ) \quad 76
\]

\[
x = "\text{A nod's as good as a wink to a blind bat.}" 
\]

\[
\text{print}( \ x \ ) \quad \text{A nod's as good as a wink to a blind bat.}
\]

\[
x = \text{int}( \ 15 / 4 \ ) - 27
\]

\[
\text{print}( \ x \ )
\]
Variables can be manipulated using expressions

```python
x = 5
print(x)  # 5
x = 7 * 9 + 13  # overwrite the previous value of x
print(x)  # 76
x = "A nod's as good as a wink to a blind bat."
print(x)  # A nod's as good as a wink to a blind bat.
x = int(15 / 4) - 27
print(x)  # -24
```
Variables manipulation

*Variables* can be manipulated using expressions
Variables can be manipulated using expressions

```python
x = 5
y = x + 7
print(x, y)
```
Variables manipulation

Variables can be manipulated using expressions

\[
x = 5 \\
y = x + 7 \\
\text{print}(x, y) \quad 5 \; 12
\]
Variables can be manipulated using expressions

```
x = 5
y = x + 7
print(x, y)  # 5 12

x = 1
print(x, y)  # 1 8
```
Variables manipulation

*Variables* can be manipulated using expressions

```python
x = 5
y = x + 7
print(x, y)  # 5 12

x = 1
print(x, y)  # 1 12
```
Variables manipulation

Variables can be manipulated using expressions

<table>
<thead>
<tr>
<th>Code</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>x = 5</code></td>
<td></td>
</tr>
<tr>
<td><code>y = x + 7</code></td>
<td></td>
</tr>
<tr>
<td><code>print(x, y)</code></td>
<td>5 12</td>
</tr>
<tr>
<td><code>x = 1</code></td>
<td></td>
</tr>
<tr>
<td><code>print(x, y)</code></td>
<td>1 12</td>
</tr>
<tr>
<td><code>x = 7*y</code></td>
<td></td>
</tr>
<tr>
<td><code>y = x + 1</code></td>
<td></td>
</tr>
<tr>
<td><code>print(x, y)</code></td>
<td></td>
</tr>
</tbody>
</table>
Variables can be manipulated using expressions
Variables can be manipulated using expressions

```plaintext
x = 5
y = x + 7
print(x, y)  # 5 12

x = 1
print(x, y)  # 1 12

x = 7*y
y = x + 1
print(x, y)  # 84 85

w = x/7
z = y/7
print(w, z)
```
Variables can be manipulated using expressions
Variables can be manipulated using expressions

```python
x=2
y=3
print("x =", x)
print("y =", y)
print("x * y =", x*y)
print("x + y =", x+y)
```
Variables can be manipulated using expressions.

```python
x=2
y=3
print( "x =", x )
print( "y =", y )
print( "x * y =", x * y )
print( "x + y =", x + y )
```
Variables manipulation

Variables can be manipulated using expressions

\[
x = 2 \\
y = 3 \\
\text{print( } "x =", x \text{ )} \\
\text{print( } "y =", y \text{ )} \\
\text{print( } "x \cdot y =", x \cdot y \text{ )} \\
\text{print( } "x + y =", x + y \text{ )}
\]
Variables can be manipulated using expressions

\[
x = 2 \\
y = 3 \\
\text{print( "x =", x ) } \\
\text{print( "y =", y ) } \\
\text{print( "x * y =", x * y ) } \\
\text{print( "x + y =", x + y ) }
\]
Variables manipulation

You may copy the contents from one variable to another

```python
x=2
y=3
print("x =", x, "and y =", y )

# Variable assignment
z=x
print("x =", x, "and y =", y )

x=y
y=z
print("x =", x, "and y =", y )
```
You may copy the contents from one variable to another

```python
x=2
y=3
print("x =", x, "and y =", y )  # Variable assignment
z=x
x=y
y=z
print("x =", x, "and y =", y )
```

```
x=2 and y=3
```
You may copy the contents from one variable to another

```python
# Variable assignment
x=2
y=3
print("x =", x, "and y =", y)

# Variable assignment
z=x
x=y
y=z
print("x =", x, "and y =", y)
```

```
x=2 and y=3
x=3 and y=2
```
Variables manipulation

You may copy the contents from one variable to another

```python
x=2
y=3
print("x =", x, "and y =", y)
# Variable assignment
z=x
x=y
y=z
print("x =", x, "and y =", y)
```

```
x=2 and y=3
Single line comment
x=3 and y=2
```
You can even use the variable itself on the right-hand side of the assignment operator (provided it was created earlier)

```python
x=2
print(x)  # x=2
x=x+3
print(x)  # x=5
```
Errors
Errors

• When something is wrong, Python rises errors

  • Syntax errors
  • Runtime Errors
    • e.g. ZeroDivisionError

• For instance, locate the problem in this program (2 statements):
  
  print( ((2*3)/4 + (5-6/7)*8 )
  
  print( ((12*13)/14 + (15-16)/17)*18 )
Variables manipulation

Spot the error....

```python
print( days_in_a_year )
days_in_a_year = 365
```
Variable names

- You are free to choose the names of your variables as you like them
- A variable name must consist of only letters, digits, and/or underscores (_)
- A variable name must start with a letter or an underscore
- Variable names are case sensitive
- A variable name should not be a reserved word

<table>
<thead>
<tr>
<th>False</th>
<th>class</th>
<th>finally</th>
<th>is</th>
<th>return</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>continue</td>
<td>for</td>
<td>lambda</td>
<td>try</td>
</tr>
<tr>
<td>True</td>
<td>def</td>
<td>from</td>
<td>nonlocal</td>
<td>while</td>
</tr>
<tr>
<td>and</td>
<td>del</td>
<td>global</td>
<td>not</td>
<td>with</td>
</tr>
<tr>
<td>as</td>
<td>elif</td>
<td>if</td>
<td>or</td>
<td>yield</td>
</tr>
<tr>
<td>assert</td>
<td>else</td>
<td>import</td>
<td>pass</td>
<td></td>
</tr>
<tr>
<td>break</td>
<td>except</td>
<td>in</td>
<td>raise</td>
<td></td>
</tr>
</tbody>
</table>
Conventions

• Be reasonable
  
  • *Never* choose variable names that are also the names of functions
  • Variable names that are in some way meaningful
  • Programmers tend to use only lower case letters in variable names
  • Multiple words name: either `useful_variable` or `usefulVariable`
  • Programmers never choose variable names that start with an underscore
Conventions

- Be reasonable
  - *Never* choose variable names that are also the names of functions
  - Variable names that are in some way meaningful
  - Programmers tend to use only lower case letters in variable names
  - Multiple words name: either `useful_variable` or `usefulVariable`
  - Programmers never choose variable names that start with an underscore

```
a = 3.14159265
b = 7.5
c = 8.25
d = a * b * b * c / 3
print(d)
```
• Be reasonable
  • *Never* choose variable names that are also the names of functions
  • Variable names that are in some way meaningful
  • Programmers tend to use only lower case letters in variable names
  • Multiple words name: either `useful_variable` or `usefulVariable`
  • Programmers never choose variable names that start with an underscore

```python
pi = 3.14159265
radius = 7.5
height = 8.25
d = a * b * b * c / 3
print(d)
```

```python
volume_of_cone = pi * radius * radius * height / 3
print(volume_of_cone)
```
Using variable for debugging

- Not always things go as you desire
- **Debugging** is the process of finding ‘bugs’
- The easiest way of debugging your code is to print the variables content in strategic places of your program
Data types

- In many programming languages you need to declare the types of your variable. For instance, in C (hard typing)
  - `int secs_per_week = 7 * 24 * 60 * 60;`

- Python does not have ‘explicit’ data typing
  - Soft typing
Data types

• In many programming languages you need to declare the types of your variable. For instance, in C (*hard typing*)

  • \texttt{int secs\_per\_week = 7 * 24 * 60 * 60;}

• Python does not have ‘explicit’ data typing

  • *Soft typing*

• You can use the function \texttt{type()} to see what the type of a variable is

\begin{verbatim}
 a=3
 print( type( a ) )
 a = 3.0
 print( type( a ) )
a = "3.0"
 print( type( a ) )
\end{verbatim}
Because of soft typing you need to pay great attention on the values that you store in a given variable over time.

```python
a = 1
b = 4
c = "1"
d = "4"
print(a + b)
print(c + d)
print(a + c)
```

Expected results?
Data types

- Because of soft typing you need to pay great attention on the values that you store in a given variable over time

```python
a=1
b=4
c = "1"
d = "4"
print( a + b )
print( c + d )
print( a + c )
```
Data types

- Because of soft typing you need to pay great attention on the values that you store in a given variable over time

```python
a=1
b=4
c = "1"
d = "4"
print( a + b )
print( c + d )
print( a + c )
```

5
Data types

- Because of soft typing you need to pay great attention on the values that you store in a given variable over time.

```python
a=1
b=4
c = "1"
d = "4"
print( a + b )
print( c + d )
print( a + c )
```

5
14
Data types

- Because of soft typing you need to pay great attention on the values that you store in a given variable over time.

```python
a=1
b=4
c = "1"
d = "4"
print( a + b )
print( c + d )
print( a + c )
```

```
5
14
Error!!!!
```
Alternative operators

number = 100
number = number + 1
print( number )

number = 100
number += 1
print( number )
Alternative operators

```
number = 100
number = number + 1
print( number )
```

```
number = 100
number += 1
print( number )
```

```
number = 100
number += 12
number -= 13
number *= 19
number /= number
print( number )
```
Comments

- Two ways to add comments to your code
  - The first is to use a hash mark (#)
    
    # comment: insert your code here
  
  - The second is to use triple double-quotes or triple single-quotes to indicate the start and end of some commentary, which may be spread over multiple lines

    """Another way of commenting on your code is via triple quotes -- these can be distributed over multiple """"
Exercises

1. Write a program that displays the number of seconds in a week.

2. The cover price of a book is €24.95, but bookstores get a 40 percent discount. Shipping costs €3 for the first copy and 75 cents for each additional copy. Calculate the total wholesale costs for 60 copies.

3. Write a program that generates the ZeroDivisionError.

4. You look at the clock and see that it is currently 14.00h. You set an alarm to go off 535 hours later. At what time will the alarm go off? Write a program that prints the answer. *Hint: for the best solution, you will need the modulo operator.*

5. Solve previous exercise 1 by assign the calculation to a variable. Then add a statement to print the contents of the variable.