Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **00**

Course Introduction

release 1.0.0

37/39 Avenue Ledru Rollin – CS 11237 – 75750 PARIS CEDEX 12 - info@valnaos.com – www.valnaos.com Tél : 01 86 95 20 48

Let's Get to Know Each Other

Introduce yourself in the following format:

- Name
- Company
- Role and background
- Familiarity with Python
- Experience in programming, programming concepts and languages
- Expectations from this course

Session Objectives

At the end of this session, you will be able to:

- Use the Python interpreter to perform simple mathematical operations.
- Output any text using Python.
- Capture text input from the user.
- Store and retrieve information from variables.
- Use numeric types `int`, `float` and `complex` and perform deeper math.
- Manipulate text data using the `str` type.
- Start using the Python interpreter on your own.

Session Syllabus

Chapter	Subject	Start	End	Total Time (in hours)
00	Course Introduction	14:00	14:30	00:30
01	Python interpreter basics	14:30	15:15	00:45
02	Data types and operators: Part I	15:15	16:00	00:45
	Coffee Break	*16:00*	*16:15*	*00:15*
03	Data types and Operators: Part II	16:15	17:00	00:45
04	Work session: Variables, numbers and strings	17:00	18:00	01:00
	Total			**04:00**

Session activities

Each session will feature a number of activities, generally scheduled in a similar fashion.
Python concepts and syntax will first be introduced theoretically with slides, followed by a quick live demo in Google Colaboratory. Each chapter will be concluded by a short exercise session reviewing the presented topics.

Finally, each class will end with a work session where students can use all the concepts they learned to solve some exercises. This is will also be a time to ask questions about the class topics, and gain deeper understanding about Python.

Q&A

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **01**

Getting to know Python

release 1.0.0

The Python Language

- Invented in 1991 by Guido van Rossum
- “Dynamically-typed, garbage-collected, interpreted language”
- Multi-paradigm: Procedural – Object-oriented – Functional
- Reference interpreter: CPython
- RIP Python 2 (2000-2020)
- Hello Python 3 ! (2008)

The Zen of Python

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. Readability counts.
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one-- and preferably only one --obvious way to do it.
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea -- let's do more of those!

The Zen of Python

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. **Readability counts.**
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. There should be one-- and preferably only one --obvious way to do it.
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea -- let's do more of those!

The Zen of Python

1. Beautiful is better than ugly.
2. Explicit is better than implicit.
3. Simple is better than complex.
4. Complex is better than complicated.
5. Flat is better than nested.
6. Sparse is better than dense.
7. Readability counts.
8. Special cases aren't special enough to break the rules.
9. Although practicality beats purity.
10. Errors should never pass silently.
11. Unless explicitly silenced.
12. In the face of ambiguity, refuse the temptation to guess.
13. **There should be one– and preferably only one –obvious way to do it.**
14. Although that way may not be obvious at first unless you're Dutch.
15. Now is better than never.
16. Although never is often better than *right* now.
17. If the implementation is hard to explain, it's a bad idea.
18. If the implementation is easy to explain, it may be a good idea.
19. Namespaces are one honking great idea -- let's do more of those!

The Python Interpreter

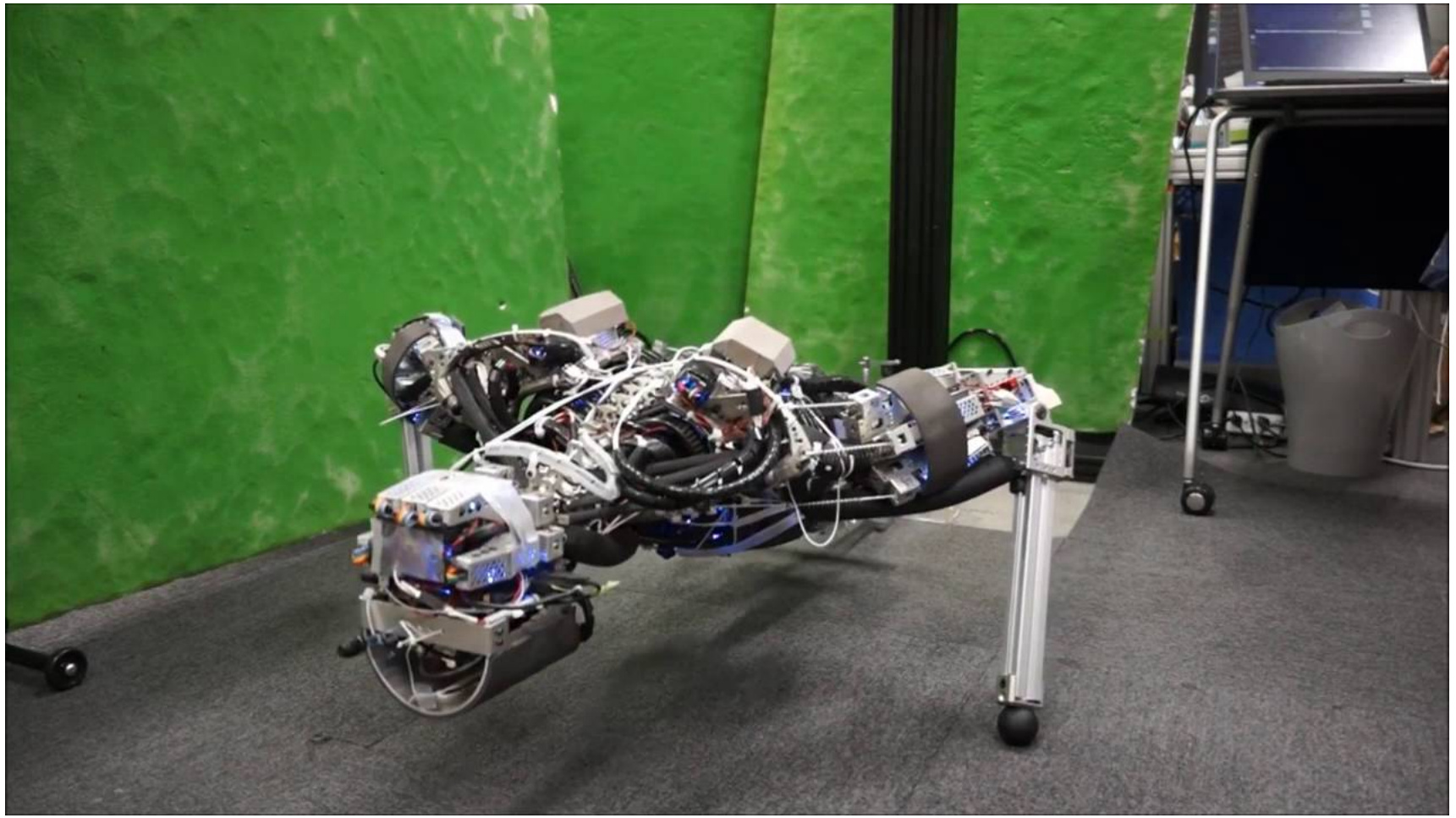
- Reference shell: `python`
- Favorite shell: IPython
- Online development environment: Google Colaboratory (https://colab.research.google.com/)

Let's write some code!

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

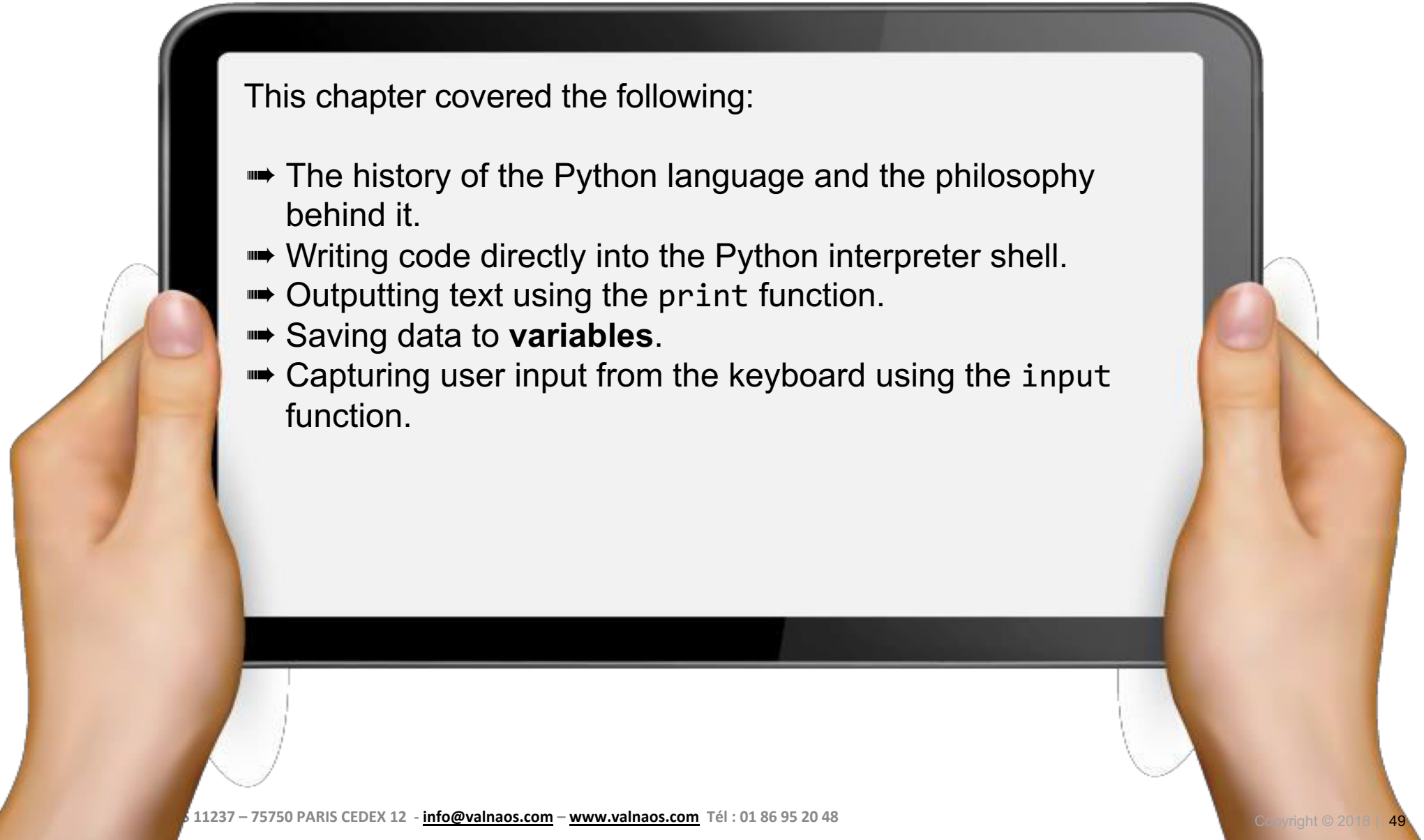
```
Hello World!
```

Workout Time!



Chapter 1 Exercises link

Chapter Summary



This chapter covered the following:

- The history of the Python language and the philosophy behind it.
- Writing code directly into the Python interpreter shell.
- Outputting text using the `print` function.
- Saving data to **variables**.
- Capturing user input from the keyboard using the `input` function.

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **02**

Data types and operators: Part I

release 1.0.0

Basic number types

- Two main numeric types in Python
  - `int` for integers
  - `float` for rational numbers (“floating point”)
- Integers have infinite precision
- Floating-point limited to IEEE-754 “double-precision” (64-bit)

Let's write some code!

```
print(1)
print(0.5)
print(1 / 2)
print(3 ** 4)
```

```
1
0.5
0.5
81
```

Basic mathematical operations

- Addition, subtraction, multiplication, division
  - `+, -, *, /`
- Python respects mathematical order or operations
  - Use parentheses to change priority
  - You can nest parentheses

```
(2 * (1 + 2)) ** 2
```

```
36
```

More mathematical operations

- Exponentiation: **
- Integer division: //

```
10 // 3
```

```
3
```

- Remainder (modulo): %

```
10 % 3
```

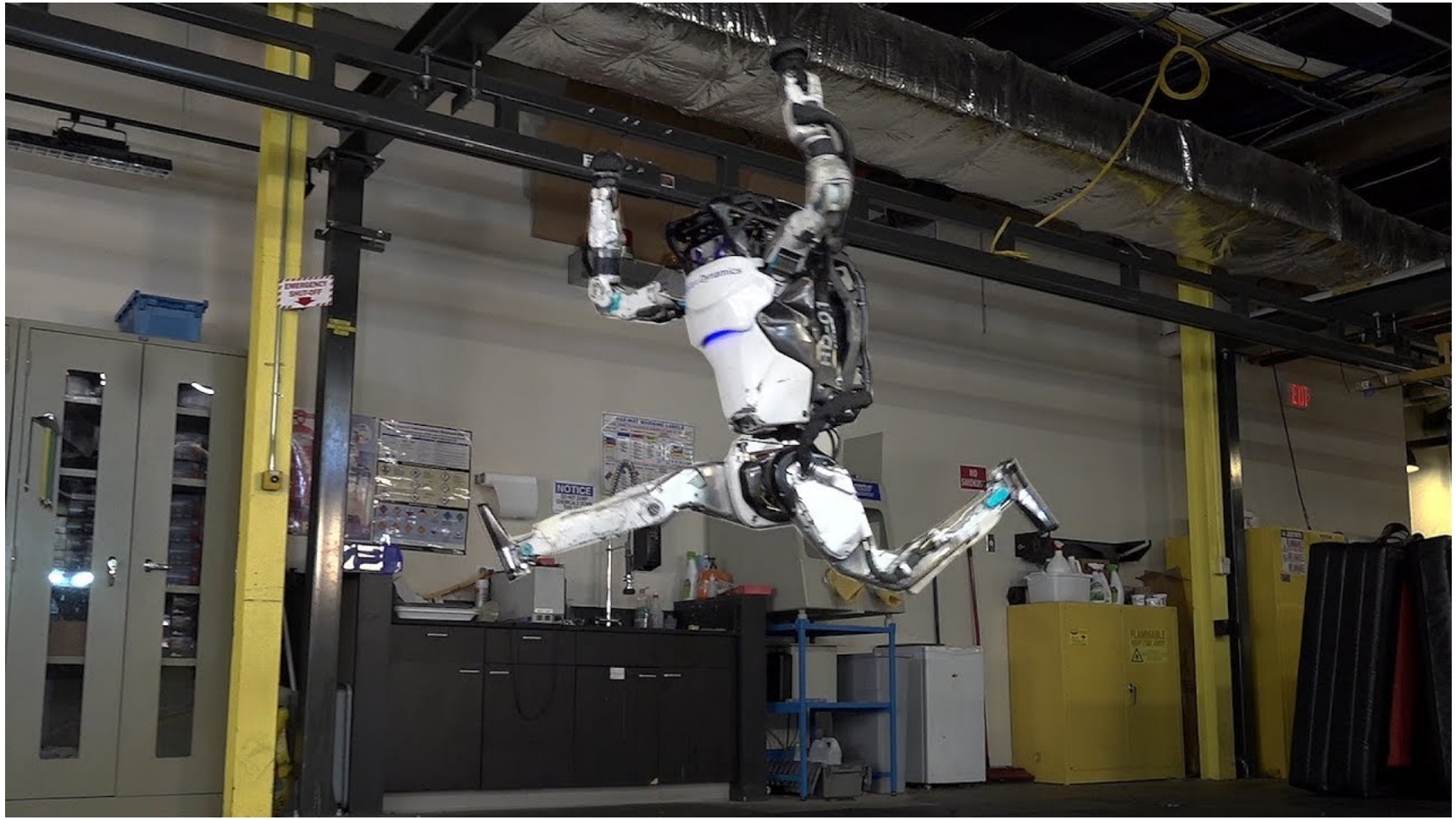
```
1
```

Deeper math

- Python has a complex type
  - 3+2j
- Absolute value: abs
- The math library contains many mathematical functions (sqrt, sin, cos…)

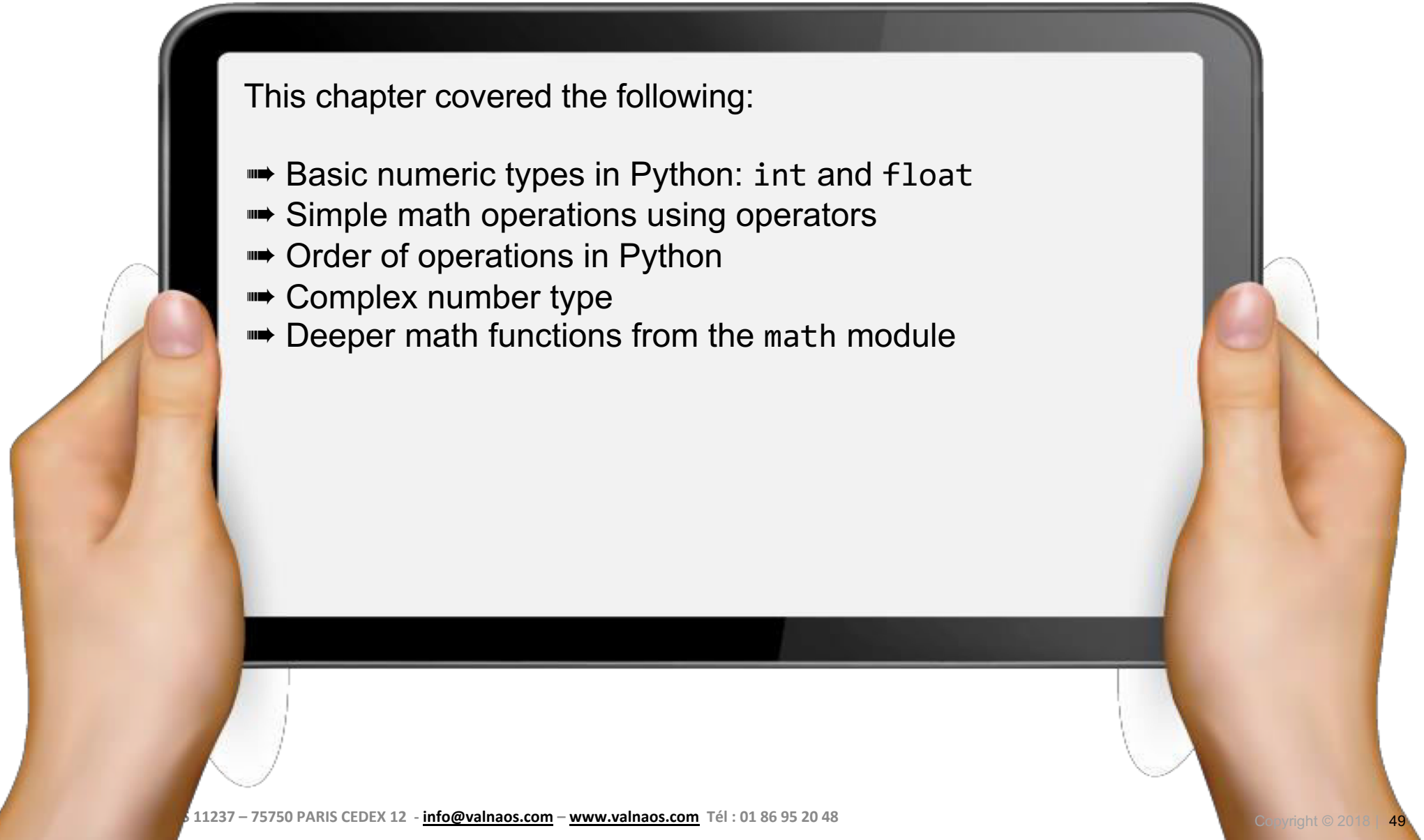
```
import math
```

Workout Time!



Chapter 2 Exercises link

Chapter Summary



This chapter covered the following:

- Basic numeric types in Python: `int` and `float`
- Simple math operations using operators
- Order of operations in Python
- Complex number type
- Deeper math functions from the `math` module

Valnaos
IT Training & Consulting

ALTRAN
.ALTRAN EDUCATION SERVICES

Python 3 Beginner

Chapter **03**

Data types and operators: Part II

release 1.0.0

37/39 Avenue Ledru Rollin – CS 11237 – 75750 PARIS CEDEX 12 - info@valnaos.com – www.valnaos.com Tél : 01 86 95 20 48

Handling text in Python

- Text data type: `str` (string)
- Strings are any arbitrary text enclosed between single or double quotes
  - `"Hello"`
  - `'World'`
- Special characters can be input using “escape sequences”
  - `\n`, `\t`, etc.
  - `\x41`, `\u262d`

Let's write some code!

```
opening_line = 'It was a bright cold day in April, and the clocks were
striking thirteen.'

print(opening_line)
```

```
It was a bright cold day in April, and the clocks were striking
thirteen.
```

Operations on Python strings

➠ Concatenation: +

```
"spam" + "eggs"
```

```
spameggs
```

➠ Repetition: *

```
5 * "spam "
```

```
spam spam spam spam spam
```

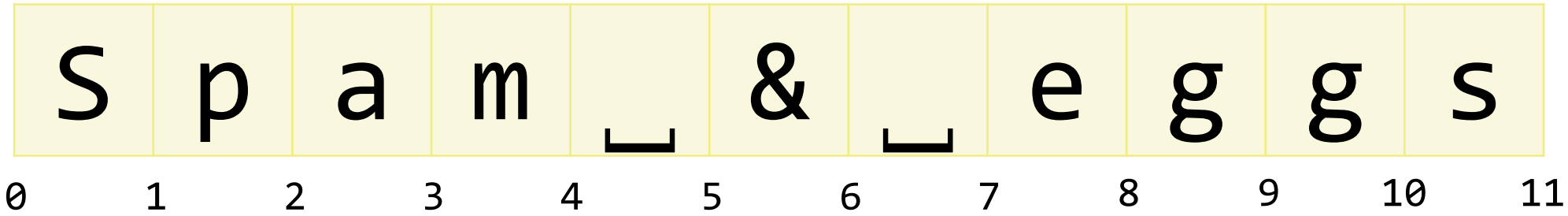
➠ Length: `len`

```
len("hello")
```

```
5
```

Anatomy of a string

A Python string is a list (string) of characters.



- The string "Spam & eggs" is shown above as an array of characters
- The length of the string is 11 characters
- Each character can be referred to by its index

Let's write some code!

```
print(3 * "spam, " + "and eggs")
```

```
spam, spam, spam, and eggs
```

The `str` type

- The `str` type has many methods.
- A method is “something that the variable can perform” (a *function* bound to an *object*).
- Change casing: `upper`, `lower`, `title`
- Change whitespace: `strip`
- Extract words: `split`

Type conversions

- A type can sometimes be converted to another type.
- Write the type name, followed by the value between parentheses:

```
a = int("12")
print(30 + a)
```

```
42
```

```
print("9 + 3 = " + str(12))
```

```
9 + 3 = 12
```

Format strings

- Format strings provide a concise and readable way to mix variables, expressions and literal strings

```
age = 12
print(f"My brother is {age} years old.")
```

```
My brother is 12 years old.
```

```
a = 12
print(f"{a} squared is {a*a}")
```

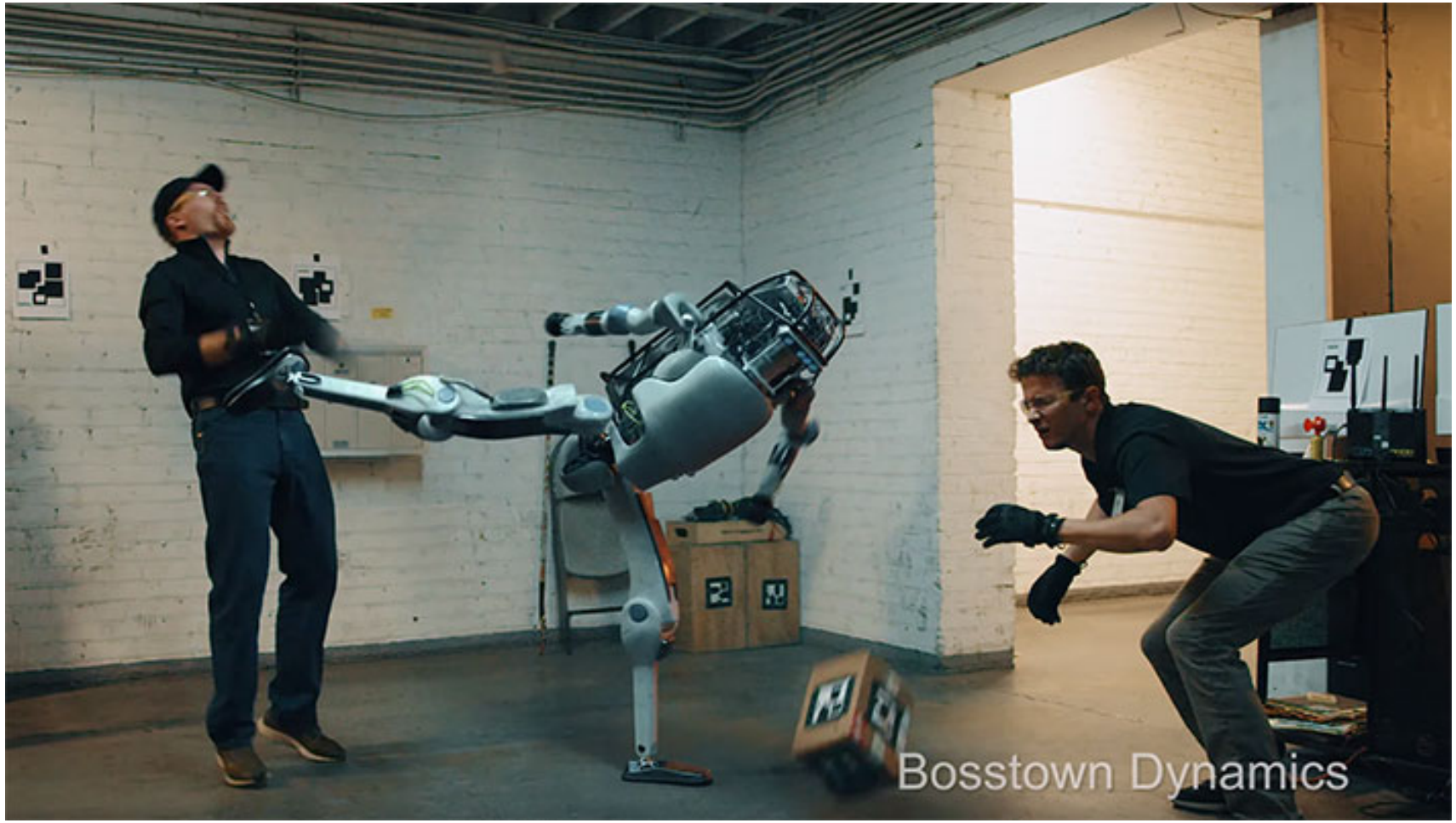
```
12 squared is 144
```

Let's write some code!

```
int("deadbeef", 16)
```

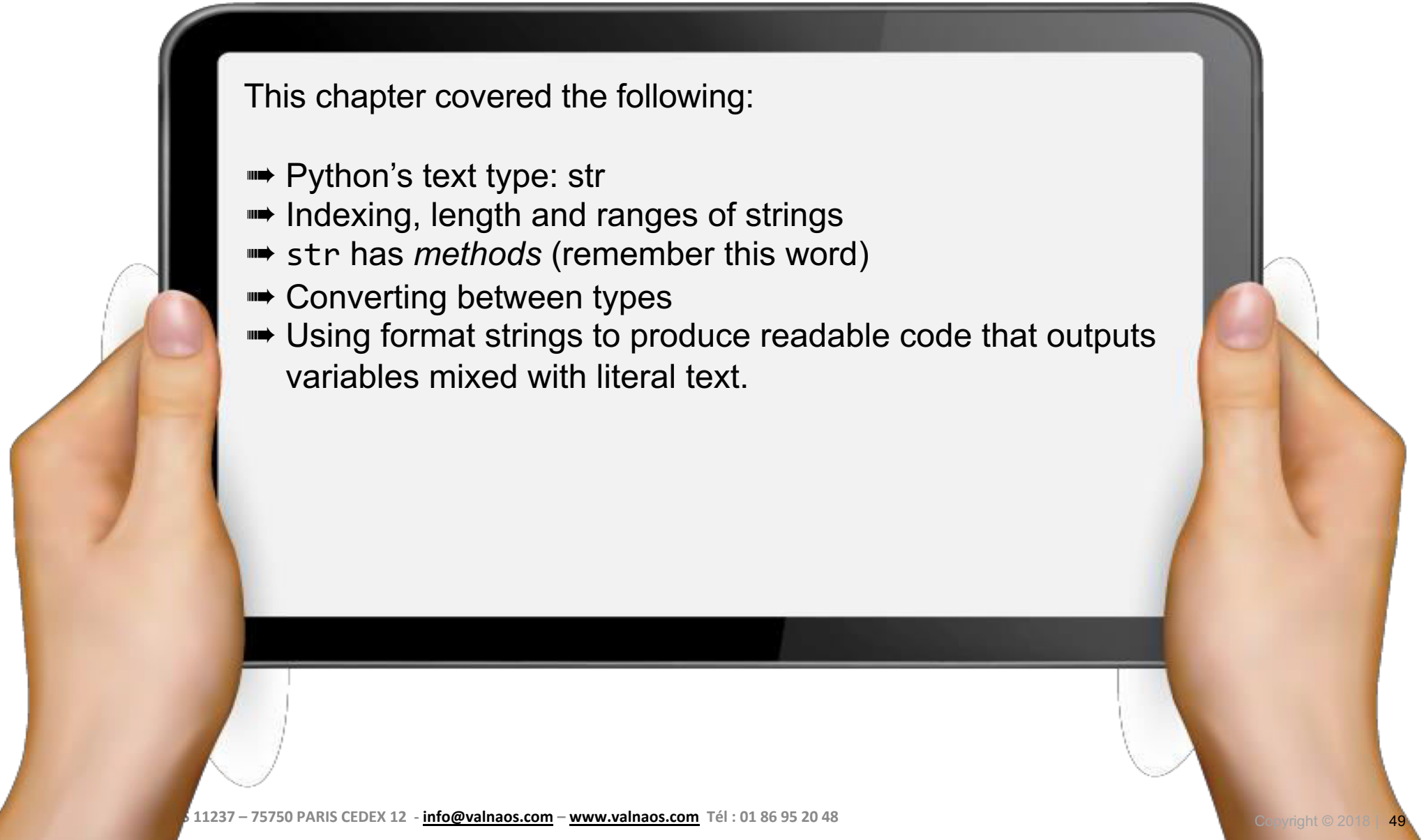
```
3735928559
```

Workout Time!



Chapter 3 Exercises link

Chapter Summary



This chapter covered the following:

- Python's text type: str
- Indexing, length and ranges of strings
- `str` has *methods* (remember this word)
- Converting between types
- Using format strings to produce readable code that outputs variables mixed with literal text.

Workshop

Session 1 Workshop link

Thank you for your attention!

