INTRODUCTION AND PYTHON BASICS

CS 3030: Python

Instructor: Damià Fuentes Escoté

UCCS University of Colorado Colorado Springs

This lesson

- Introduction
- Course description
- Syllabus
- Python basics

About me


Canada
Greenland
Hudson Bay
Labrador Sea
United States
Mexico
Gulf of Mexico
North Atlantic Ocean
Caribbean Sea
San Francisco
Los Angeles
San Diego
Dallas
Mexico City
Ottawa
Montreal
Toronto
Philadelphia
Cuba
Guatemala
Honduras
Nicaragua
Dominican Republic
Puerto Rico
Romania
Hungary
Serbia
Croatia
Greece
Italy
Rome
France
Paris
Spain
Barcelona
Tyrrhenian Sea
Ireland
United Kingdom
Netherlands
Belgium
Austria
Berlin
North Sea
Syria
Lebanon
Jordan
Israel
Egypt
Libya
Tunisia
Algeria
Morocco
Western Sahara
Mauritania
Mali
Niger
Nigeria
Burkina Faso
Gabon
Ghana
Senegal
Dakar
Guinea
Côte d'Ivoire
Liberia
Satellite
Google

About me


Canada
United States
Mexico
San Francisco
Los Angeles
San Diego
Dallas
Mexico City
Ottawa
Montreal
Toronto
Philadelphia
Hudson Bay
Gulf of Mexico
Cuba
Guatemala
Honduras
Nicaragua
Dominican Republic
Puerto Rico
Satellite
Romania
Hungary
Syria
Lebanon
United Kingdom
Netherlands
Belgium
Austria
Serbia
Croatia
Greece
Jordan
Israel
Ireland
Paris
France
Italy
Rome
Tyrrhenian Sea
Barcelona
Egypt
Tunisia
Spain
Libya
Algeria
Morocco
Niger
Western Sahara
Mali
Nigeria
Mauritania
Burkina Faso
Gabon
Ghana
Senegal
Côte d'Ivoire
Dakar
Guinea
Liberia
UNIVERSITAT POMPEU FABRA
MonkingMe
LANZADERA
Persigue tu sueño emprendedor

About me


Balsells
Fellowship Programs
Balsells Foundation—University of California—University of Colorado—Generalitat de Catalunya
UCCS
University of Colorado
Colorado Springs
Canada
Greenland
Hudson Bay
Labrador Sea
United Kingdom
Ireland
France
Spain
Italy
Rome
Tyrrhenian Sea
Greece
Romania
Hungary
Austria
Serbia
Croatia
Belgium
Netherlands
Berlin
Paris
Syria
Lebanon
Jordan
Israel
Egypt
Libya
Tunisia
San Francisco
Los Angeles
San Diego
Dallas
Mexico
Mexico City
Gulf of Mexico
Cuba
Guatemala
Honduras
Nicaragua
Dominican Republic
Puerto Rico
Nigeria
Gabon
Liberia
Ottawa
Montreal
Toronto
Satellite
Google

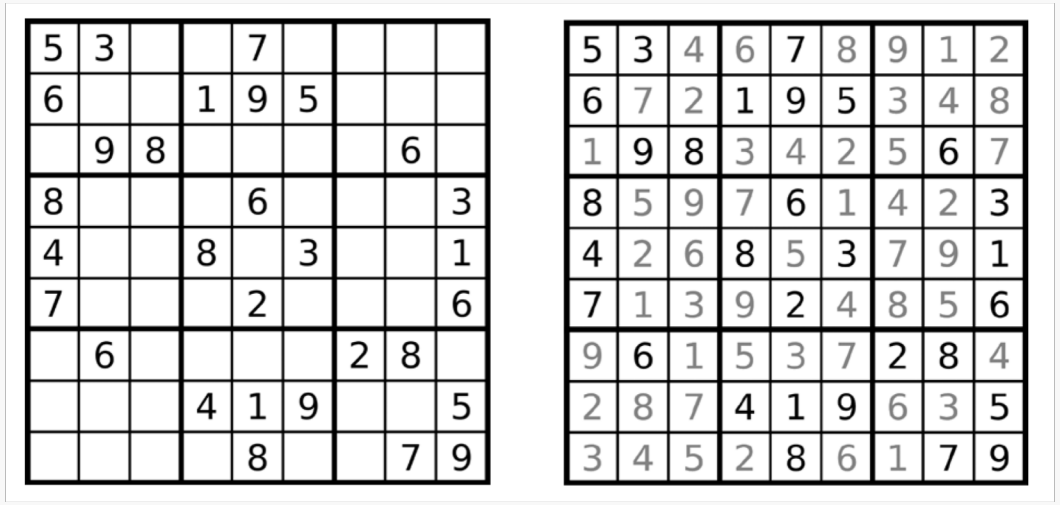
Why this course?

- Python is:
  - *High-level programming language*
  - *Active and supportive community*
  - *Extensive support libraries*
  - *Reliable and efficient*
  - *Accessible (easy to learn and use)*
  - *High demanded by companies*

- With python you are able to solve real-world problems with just a few concepts!

Programmers don’t need to know that much math

- Just because Sudoku involves numbers doesn’t mean you have to be good at math to figure out the solution. The same is true for programming.



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

5	3	4	6	7	8	9	1	2
6	7	2	1	9	5	3	4	8
1	9	8	3	4	2	5	6	7
8	5	9	7	6	1	4	2	3
4	2	6	8	5	3	7	9	1
7	1	3	9	2	4	8	5	6
9	6	1	5	3	7	2	8	4
2	8	7	4	1	9	6	3	5
3	4	5	2	8	6	1	7	9

- The more you program, the better you’ll become
- Programming is a creative task!

ISBN-10: 1-59327-599-4
ISBN-13: 978-1-59327-599-0

Course content

- Python basics
- Advanced python concepts
- Examples of interview exercises
- Pattern matching with regular expressions
- Reading, writing and organizing files
- Web scraping
- Working with excel, word, pdf, csv and json files and data
- Keeping time, scheduling tasks, and launching programs
- Sending email and text messages
- Network fundamentals and socket programming (TCP/UDP)
- Scientific computing with NumPy
- Data visualization with matplotlib

Not everything in the book!

Course content examples – Python basics

- ```
  name = ''
  while name != ‘Damia':
      print('Please type your name.')
      name = input()
  print('Thank you!')
  ```
  ```

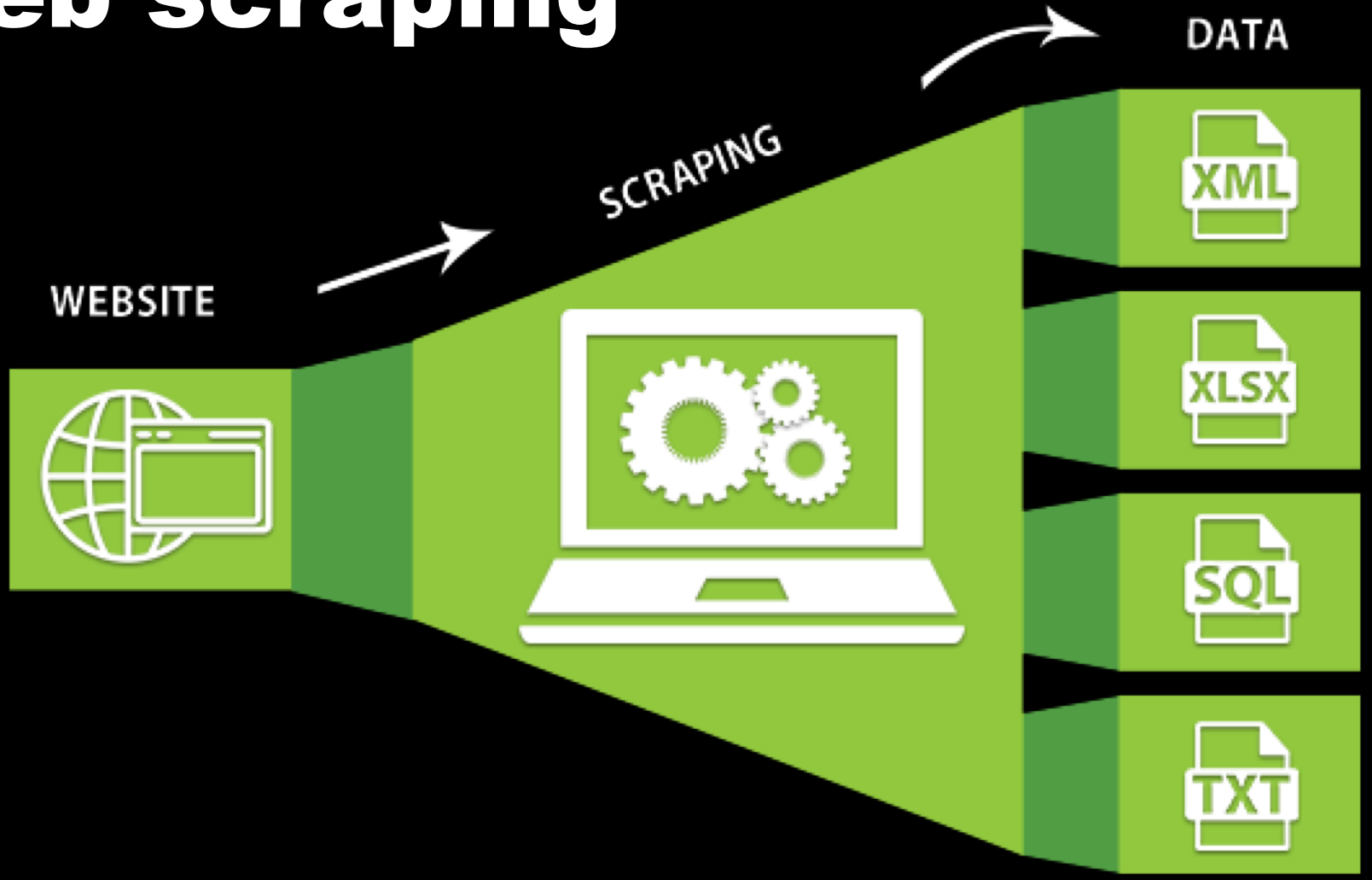
Course content examples – Advanced python

- *def my_decorator(func):*
  def wrapper():
  print("Something is happening before the function is called.")
  func()
  print("Something is happening after the function is called.")
  return wrapper

  @my_decorator
  def say_whee():
  print("Whee!")

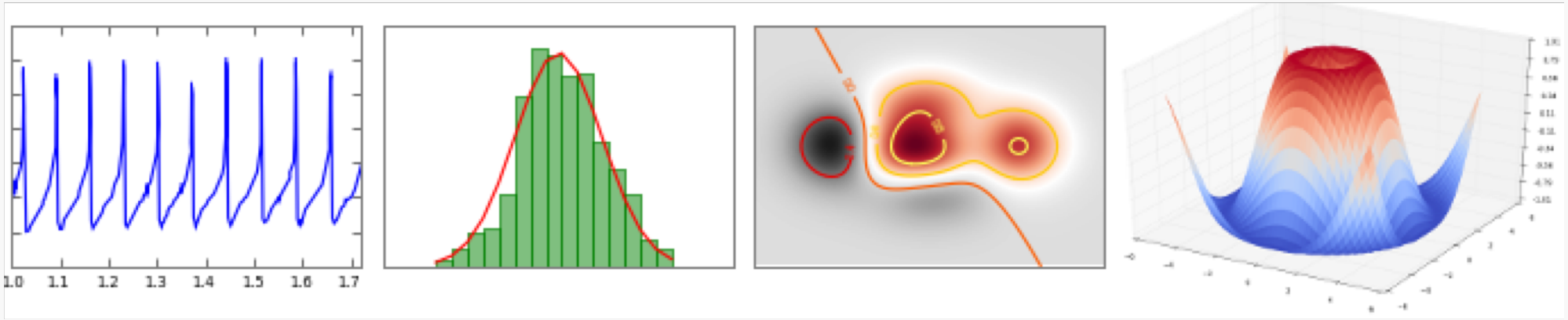
Course content examples – Interview exercises

Course content examples – Web scraping



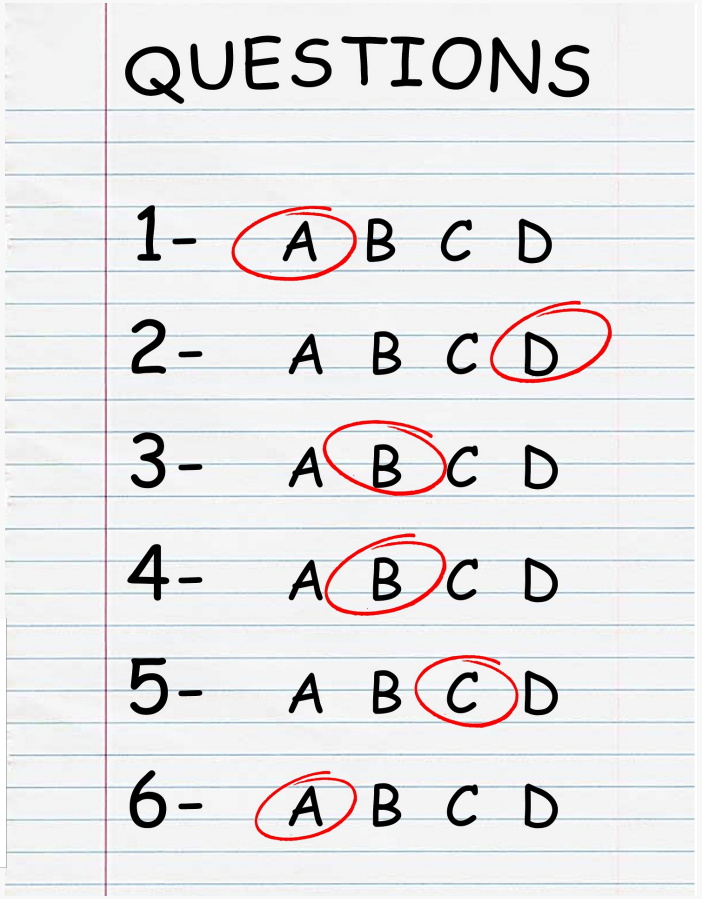
Course content examples – Send email and text messages

Course content examples – Data visualization with matplotlib



Quizzes

- Lessons topics
- Random days
- 5-10 minutes quiz of the past topics
- 10% of the mark
- Individual. Laptop not required.



4. What is printed by the Python code? Be careful: Note the backslashes:

```
print('how\nis it\nnow')
```

```
4. how
   is it
   now
```

Homework

Homework	Given	Due
Homework 1	January 22nd	Feb 3rd
Homework 2	January 29th	February 10th
Homework 3	February 5th	February 17th
Homework 4	February 12th	February 24th
Homework 5	February 19th	March 3rd
Homework 6	February 26th	April 10th
Homework 7	March 5th	March 17th
Homework 8	March 12nd	March 24th
Homework 9	April 2nd	April 21st
Homework 10	April 18th	May 5th

- 30% of the mark
- Individual.

Midterm

- March 21st – **Midterm**, 10% of the mark
- Individual.
- In-class.
- Laptop required.
- Exercises similar to quizzes and homework.

Interview exercises

- Examples of **interview exercises** asked by Google, Amazon, etc
- April 25th and 30th
- Individual.
- In-class.
- Laptop recommended.

Final exam

- Topics about lessons, interview exercises, quizzes and homework.
- May 9th – **Final exam**, 10% of the mark
- Individual.
- In-class.
- Laptop required.

Final project

- 40% of the mark
- Develop whatever you want! But in python.
- Groups of 2 or 3.
- One-page project proposal by March 21st
  - *You can send me the proposal before and start working once I accept it.*
- Demo and presentation between May 2nd - 16th
- 5 – 10 page paper

Grading

Category	Percentage
Homework	30%
Quizzes	10%
Final project	40%
Midterm	10%
Final exam	10%

Letter grade distribution			
>= 93.00	A	**73.00 - 76.99**	C
90.00 - 92.99	A-	**70.00 - 72.99**	C-
87.00 - 89.99	B+	**67.00 - 69.99**	D+
83.00 - 86.99	B	**63.00 - 66.99**	D
80.00 - 82.99	B-	**60.00 - 62.99**	D-
77.00 - 79.99	C+	**<= 59.99**	F

Cheating

- Plagiarism is a serious offense and all involved parties will be penalized according to the Academic Honesty Policy.
- I will report to the Dean's Office and they will decide.
  - *I will not be able to control the consequences.*
- If you copy something like a method, a class, etc. reference it!

Let’s start talking about python!

Python interpreter

- An interpreter is a program that reads and executes code.
- Python interpreter software reads source code (written in the Python language) and performs its instructions.

Source code

```
passwordFile = open('SecretPasswordFile.txt')
secretPassword = passwordFile.read()
print('Enter your password.')
typedPassword = input()
if typedPassword == secretPassword:
   print('Access granted')
   if typedPassword == '12345':
      print('That password is one that an idiot puts on their luggage.')
else:
   print('Access denied')
```

Interactive shell

- A shell is a program that lets you type instructions into the computer.
- Python's interactive shell lets you enter instructions for the Python interpreter software to run. The computer reads the instructions you enter and runs them immediately.
- From the Command Line (Windows) or Terminal (Mac) write:

  $ python

  >>> print('Hello world!')

  Hello world!

Virtual Environments

- What is a Virtual Environment?
  - Create an isolated environment for Python projects
  - Each project can have its own dependencies

- Why?
  - *Don't install all those packages (or modules) to your system*

- How?

  $ pip install venv *(or virtualenv)*

  $ python3 -m venv env

  $ source env/bin/actívate

  (env) $ deactivate

Try

$ which python

(env) $ which python

Python Software

They use the python interpreter!

Recommended:

- PyCharm Community Edition IDE
  - *https://www.jetbrains.com/pycharm/*

Others:

- Visual Studio IDE
- Sublime text + Command line
- IDLE (used in the book)
- Online Python Editor and IDE (https://repl.it/repls/ProfitableBurlywoodBracket)
- etc

Hardware
Any computer!

Python basics

- **Expression**
  - *Combinations of **values** and **operators** that can **evaluate** down to a single value*
  - *2 + 2*
  - *4 * 3*
  - *5 – 4 * (3 + 1)*
  - *2*

Python basics

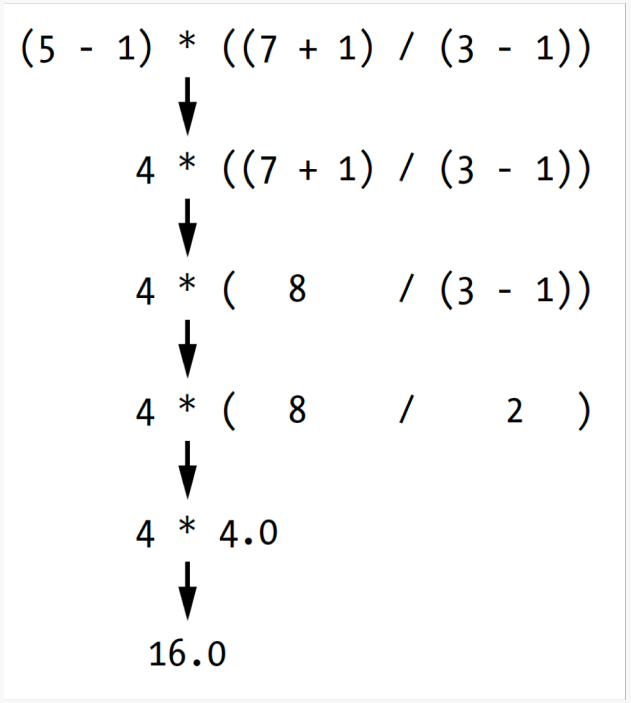
Operator	Operation	Example	Evaluates to...
**	Exponent	`2 ** 3`	8
%	Modulus/remainder	`22 % 8`	6
//	Integer division/floored quotient	`22 // 8`	2
/	Division	`22 / 8`	2.75
*	Multiplication	`3 * 5`	15
-	Subtraction	`5 - 2`	3
+	Addition	`2 + 2`	4

Python basics

- **Precedence**
  - *The ** operator is evaluated first;*
  - *the * , / , // , and % operators are evaluated next, from left to right;*
  - *the + and - operators are evaluated last (also from left to right).*
  - *You can use parentheses ( ) to override the usual precedence if you need to.*

Python basics

- **Precedence**
  - *Python will keep evaluating parts of the expression until it becomes a single value*



Python basics

- **Common data types**
  - *A data type is a category for values, and every value belongs to exactly one data type*
  - *The meaning of an operator may change based on the data types of the values next to it.*
  - *String replication operator*
    - 'Alice' * 5

Table 1-2: Common Data Types

Data type	Examples
Integers	`-2, -1, 0, 1, 2, 3, 4, 5`
Floating-point numbers	`-1.25, -1.0, --0.5, 0.0, 0.5, 1.0, 1.25`
Strings	`'a', 'aa', 'aaa', 'Hello!', '11 cats'`

Storing values in variables

- You'll store values in variables with an **assignment statement**. An assignment statement consists of a variable name, an equal sign (called the **assignment operator**), and the value to be stored.
- A variable is **initialized** (or created) the first time a value is stored in it
- **Overwriting** the variable -> When a variable is assigned a new value, the old value is forgotten. It doesn't matter the data type.
- **Multiple assignment**
  - *a = b = c = 1*
  - *a,b,c = 1,2,"john"*

Variable names

- 3 rules:
  - *It can be only one word.*
  - *It can use only letters, numbers, and the underscore (_) character.*
  - *It can't begin with a number.*

Valid variable names	Invalid variable names
`balance`	`current-balance` (hyphens are not allowed)
`currentBalance`	`current balance` (spaces are not allowed)
`current_balance`	`4account` (can't begin with a number)
`_spam`	`42` (can't begin with a number)
`SPAM`	`total_$um` (special characters like `$` are not allowed)
`account4`	`'hello'` (special characters like `'` are not allowed)

Variable names

- Variable names are **case-sensitive**
  - *spam , SPAM , Spam , and sPaM are four different variables.*
- It is a Python convention to start your variables with a lowercase letter.
- Descriptive names makes your code more readable.
- **Camelcase**
  - *lookLikeThis*
- **Underscores** (official Python code style)
  - *looking_like_this*
- It doesn’t matter which way, but be consistent in your code!

First program

- Ask the user for his name and age.
- Tell him:
  - *The length of his name*
  - *How old will be in a year (= age + 1)*

- https://repl.it/languages/python3

First program

- The **print()** function displays the string value inside the parentheses on the screen.
- The **input()** function waits for the user to type some text on the keyboard and press enter. Always returns a string.
- The **len()** function evaluates to the integer value of the number of characters in the string argument (or a variable containing a string).
  - *len('hello world') evaluates to 11*
- The **str()** function can be passed an integer value and will evaluate to a string value version of it.
  - *str(29) evaluates to '29'*
- The **int()** function can be passed an string value and will evaluate to a int value version of it.
  - *int('29') evaluates to 29*

Comments

- Python ignores comments, and you can use them to write notes or remind yourself what the code is trying to do.
- Any text for the rest of the line following a hash mark (#) is part of a comment.
  - *# This is a one line comment*
- Any text in between two delimiters (''') is part of a comment. Used to explain things in more detail.
  - '''

    This is a multiline comment.

    Second line.

    '''