

# Programming Thinking

Pepe García



SCHOOL OF  
SCIENCE &  
TECHNOLOGY

# Programming Thinking



SCHOOL OF  
SCIENCE &  
TECHNOLOGY

# Plan for today

Lists

For loops



SCHOOL OF  
SCIENCE &  
TECHNOLOGY

# Lists

Lists are sequences of values



# Constructing Lists

We construct lists with the brackets `[]` syntax:

```
[1, 2, 3, 4, 5]
```

```
["hello", "dolly"]
```

```
[]
```

```
[1, "hello", 2, "dolly", 3]
```



# Constructing Lists



# Accessing list elements

We use **square brackets** to access elements by their **index**.

**indices** in lists start by **0**, not 1.

```
words = ["hello", "dolly"]  
words[0]  
# "hello"  
words[1]  
# "dolly"
```



# Accessing List elements





# Operators on lists

As with strings, `+` and `*` operators work with lists too!



# Operators on lists



SCHOOL OF  
SCIENCE &  
TECHNOLOGY

# Mutating lists

Lists are mutable values, and they provide functionality to add, delete, and update elements



# Updating elements in the list

To update an element inside the list, we use a syntax similar to the one for declaring variables, but using the brackets and the index we refer to.

```
numbers = [1,2,4]
numbers[2] = 3
print(numbers)
```

# Updating elements in the list



# Appending elements to the list

To add a new element to the end of the list we use the `append()` method on it.

```
numbers = [1,2,3]
numbers.append(4)
print(numbers)
```



# Appending elements to the list



# Inserting elements in the list

There's an alternative way of adding new elements to the list, and it's using the `insert()` method on it:

```
words = ["hello", "my", "friends"]  
words.insert(2, "dear")  
print(words)
```

The difference between this and `append` is that with `insert` we can choose where to put it by using the target index





# Inserting elements in the list



# Removing elements from the list

In order to remove an element from a list, we should use the `.pop()` method, and pass the index of the element we want to remove

```
words = ["hello", "my", "friend"]  
words.pop(1)  
print(words)
```



# Removing elements from the list



# For loops

For loops are simpler than while loops. They iterate over elements in a list.



# For loops

```
for <variable_name> in <list>:  
    <body>
```



# For loops



SCHOOL OF  
SCIENCE &  
TECHNOLOGY

# Recap

Use lists to store collections of values

Use mutation operations on list to add, remove, or update elements in the list

Use for loops to iterate over elements in the list



# Exercises

1. Create a function that returns a list of numbers from 0 to 10
2. Create a function that takes a list of numbers (you can use the list you created in the previous exercise) and returns the sum of them
3. Investigate the `~range()~` function. After you've used it, create a function that receives a number as parameter and prints all the numbers from it to zero (using a for loop).
4. Create a function that takes a list of numbers and returns the maximum value among them
5. Create a function that takes a list of numbers and returns the minimum value among them



# Recommended literature

<https://www.py4e.com/html3/05-iterations>

<https://www.py4e.com/lessons/lists>



SCHOOL OF  
SCIENCE &  
TECHNOLOGY