Lists

In this lesson, we will learn about an important data structure in Python: lists.

WE'LL COVER THE FOLLOWING

A **data structure** is a way of storing and organizing data according to a certain format or structure. Data structures are a crucial part of computer programming. Since we frequently deal with data manipulation, it is of paramount importance to organize it in an efficient and meaningful way.

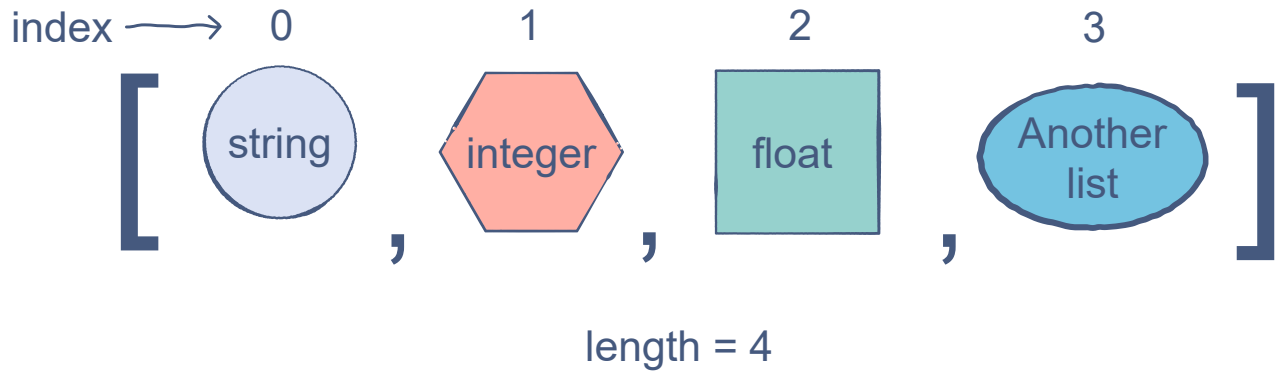
The three commonly used built-in data structures offered in Python 3 are:

1. List
2. Tuple
3. Dictionary

List

List allows us to store elements of different data types in one container. The contents of a list are enclosed by square brackets, `[]`.

Lists are ordered and elements are stored linearly at specific indexes.



Creating

```
science = ["Physics", "Chemistry", 30]
print (science)

# Indexing is done using square brackets
print (science[0])

# Length
print (len(science))
```

Here's an example of lists inside another list:

```
world_cup_winners = [[2006, "Italy"], [2010, "Spain"], [2014, "Germany"], [2018, "France"]]
print (world_cup_winners)
print (world_cup_winners[1])
print (world_cup_winners[1][0])
```

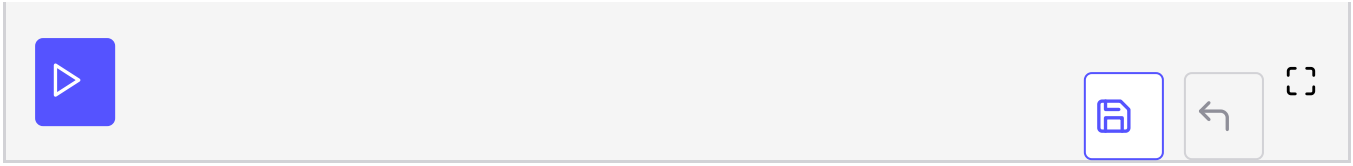
Merging

Python makes it really easy to merge lists together. The simplest way is to use the `+` operator or the `extend()` function:

+_operator | extend()

```
part_A = [1, 2, 3, 4, 5]
part_B = [6, 7, 8, 9, 10]
merged_list = part_A + part_B
print (merged_list)
```

```
print (merged_list)
```

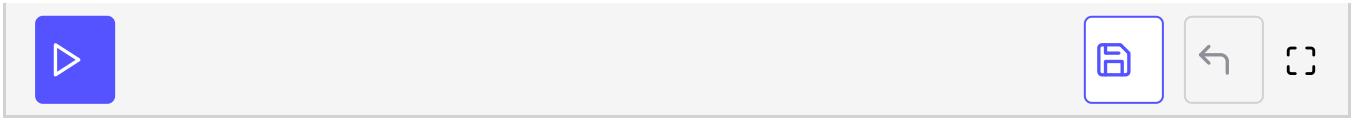


Appending

The `append()` function appends the input argument at the end of the list.

```
list.append(value)
```

```
lst = [1, 3, 5]
print(lst)

# print after append
print("List after append:")
lst.append(7)
print(lst)
```



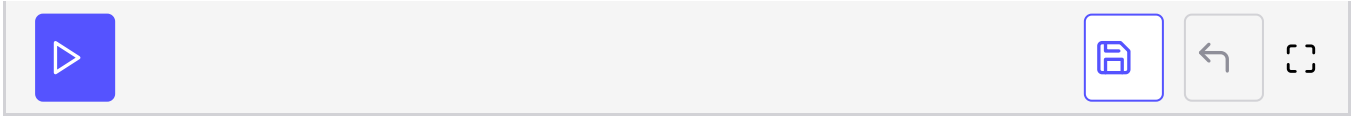
List Comprehension

List comprehension is a technique that uses a `for` loop and a condition to create a new list from an existing one.

A list comprehension statement is always enclosed in square brackets, `[]`. Let's see how list comprehension makes our task easier:

without_comprehension | with_comprehension

```
nums = [1, 2, 3, 4, 5]
nums_square = []

for n in nums:
  nums_square.append(n ** 2)

print (nums)
print (nums_square)
```

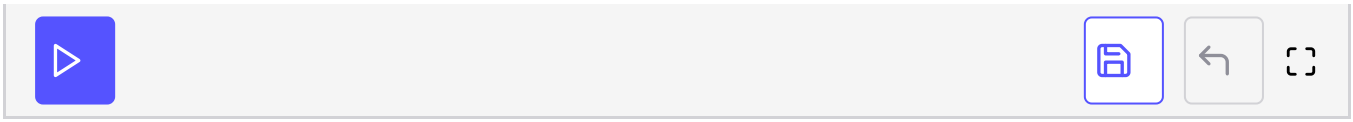


List comprehension can also be performed on more than one list. The number of `for` loops in the comprehension will correspond to the number of lists

of `for` loops in the comprehension will correspond to the number of lists we're using.

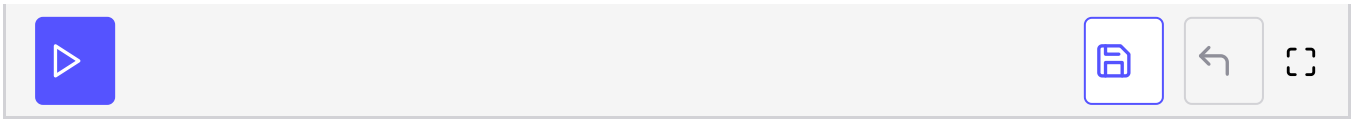
Below is an example of creating a list of lists using two `for` loops:

```
M = [[], [], [], [], []]

for m in range(5):
  for n in range(5):
    M[m].append(n + m * 20)

print(M)
```



Now let's perform the same task using list comprehension:

```
M = [[(n + m * 20) for n in range(5)] for m in range(5)]

print(M)
```



In the next lesson, we will learn about tuples and dictionaries.