LEARN CODING

ale66

DATA STRUCTURES

ABSTRACTION

In CS all efforts are made to create -abstractions- of computers/memory that let us program them easily and in a -portable- way

Data structures are presentations of our data that support our coding

SO FAR

- variables as name/type/value triples

- a version of Spreadsheet cells
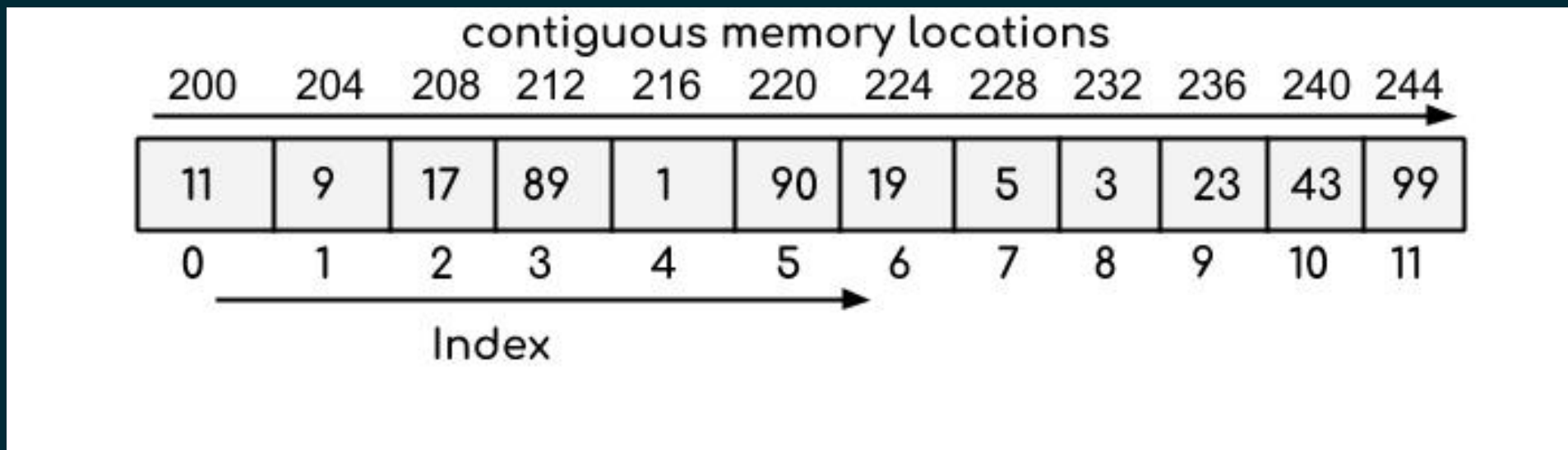
- *scalar* or *atomic* datum

TODAY

- variables as name/type/structure/values

- a version of spreadsheet columns (or rows)

- multi-dimensional data

- an indexing systems allows to reach each single value

- The way in which the data is organized guides coding and affects computation

- data structures are a *design choice* based on:

  - the nature of the data

  - the processes that need to be performed

- Python offers simple data structures, their adoption affects code readability as well as scalability.
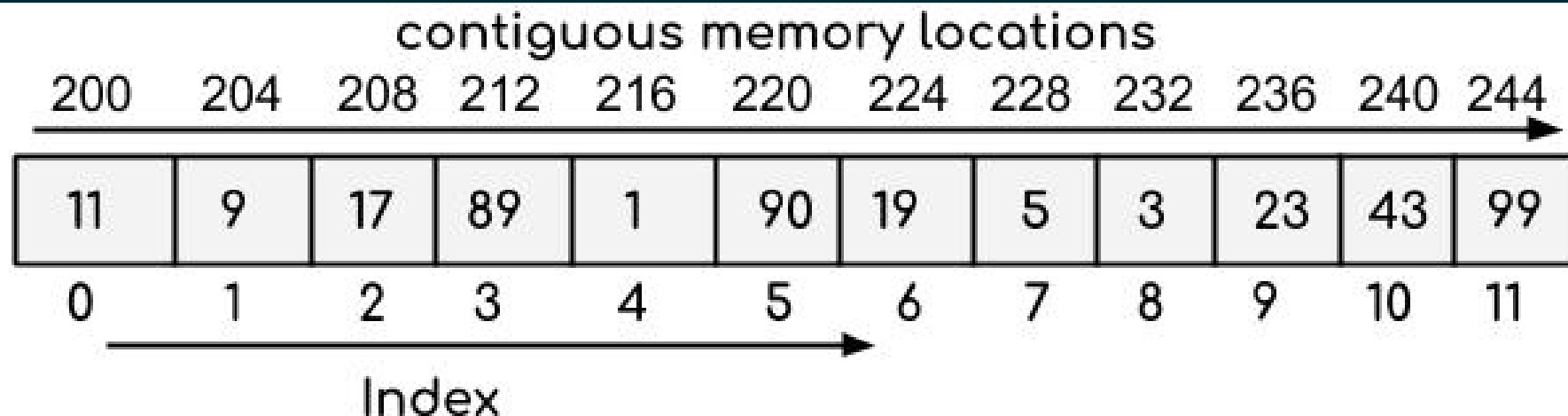
LISTS

a sequence (order matters) which is thought of as a contiguous block of memory locations, each of them storing a fixed-length data item.

contiguous memory locations

200	204	208	212	216	220	224	228	232	236	240	244
11	9	17	89	1	90	19	5	3	23	43	99
0	1	2	3	4	5	6	7	8	9	10	11

Index

the most common data structure in Python

- used to store multiple, oft. homogeneous data in a single variable

- any data type and combinations of data types

- the elements of a list are indexed

- such indexing starts from 0

OBSERVATIONS

lists

- Can be ordered

- Changeable

- Support duplicated members

QUIZ 1! TRUE OR FALSE?

SOLUTIONS

- A Python list is an ordered and changeable collection of data where we can only store data of the same data type.

  - **False, we can store data of different types**

- We use the index of a list to access elements, and this index number is the actual address of a memory block.

  - **False, indexes represent the memory addresses, but always start from 0**

- Since lists are indexed, lists can have items with the same value.

  - **True!**

WORK WITH LISTS

github.com/ale66/learn-coding

A LIST OF STRINGS

A simple list of London football clubs

```
1 teams = ['Chelsea FC', 'Arsenal FC', 'Crystal Palace FC', 'West Ham FC']
```

```
1 len(teams)
2 4
```

ACCESS BY INDEX

```
1  print(teams[0])
2
3  Chelsea FC
```

```
1  print(teams[3])
2
3  West Ham FC
```

```
1  if (teams[0] == 'Chelsea FC'):
2
3          print('Come on Chelsea!')
```

MUTABLE VALUES

```
1  fruitlist = ["apple", "banana", "cherry"]
2
3  print(fruitlist)
```

```
1  # Update a value
2  fruitlist[1] = 'blackcurrant'
3
4  print(fruitlist)
```

STRINGS

PYTHON STRINGS ARE LISTS

```
1  mystring = 'python'
```

	p	y	t	h	o	n
	0	1	2	3	4	5
	-6	-5	-4	-3	-2	-1

```
1  print(mystring[0])
```

```
1  print(mystring[-1])
```

SLICING

We can access arbitrary segments of the list/string:

```
1  alist = [1, 3, 5, 7, 9, 11]
2
3  print(alist[0:2])
```

Important: intervals are closed on the left and open on the right:

the element in position 0 is printed but that in position 2 ain't

QUIZ 2! TRUE OR FALSE?

```
1  mydata = [12, 32, 1, 43, 65]
```

- a. will `print(mydata[3])` print 1?

- b. will `print(mydata[1]+18)` print 50?

- c. will `print(mydata[-1])` print 65?

- d. will `print(mydata[0:3])` print [12, 32, 1]?

github.com/ale66/learn-coding

SOLUTIONS

- a: False

- b: True

- c: True

- d: True

ITERATIONS

ITERABLES

Python lists are called *iterables* because they are likely subjects or the repetition (iteration) of a fixed sequence of instructions

```
1  Print the content of the string varible vertically, i.e., each letter of th
```

```
1  for letter in mystring:
2      print(letter)
3      # a 'newline' is automatically emitted at the end of each print
```

As with `if`, the 4-spaces indentation defines the block of code to be exectued zero, one or many times.

UNHELPFUL...

```
1  print(mystring[0])
2  print(mystring[1])
3  print(mystring[2])
4  ...
```

Inflation! Raise all prices in the menu by 10pc

```
1  alist = [2, 3, 5, 7]
2
3  # fix me!
4  for price in alist:
5      price = price*1.1
6      print(f'The new price is {price}')
```

We need to amend each element of the existing list, as in

```
1      alist[0] = alist[0]*1.1
```

but when to stop?

RANGE

A function that generates the indices needed to amend the
list element by element

```
1  range(5) = 0, 1, 2, 3, 4
```

```
1  range(3, 6) = 3, 4, 5
```

```
1  range(3, 8, 2) = 3, 5, 7 # +2 at each step
```

```python
1  alist = [2, 3, 5, 7]
2
3  howmany = len(alist)
4
5  for i in range(howmany):
6      # prices are changed forever
7      alist[i] = alist[i]*1.1
```

EXERCISE:

```
1 teams = ['Chelsea FC', 'Arsenal FC', 'Crystal Palace FC', 'West Ham FC']
```

print out the club names without the FC suffix

COLLABORATIVE CODING (PAIR CODING)

Get the VS Code extension:

Live Share

(experimental) Sign in with your unimit.it account as if it were Microsoft

create and share tokens to work on the same file (via third-party hosting)

EXERCISE, A

Print available fruits but not bananas

```
1  fruits = ['apple', 'banana', 'cherry', 'blackcurrant']
```

hint: use `continue`

EXERCISE, B

Print available fruits but stop as soon as you find bananas

```
1  fruits = ['apple', 'banana', 'cherry', 'blackcurrant']
```

hint: use break