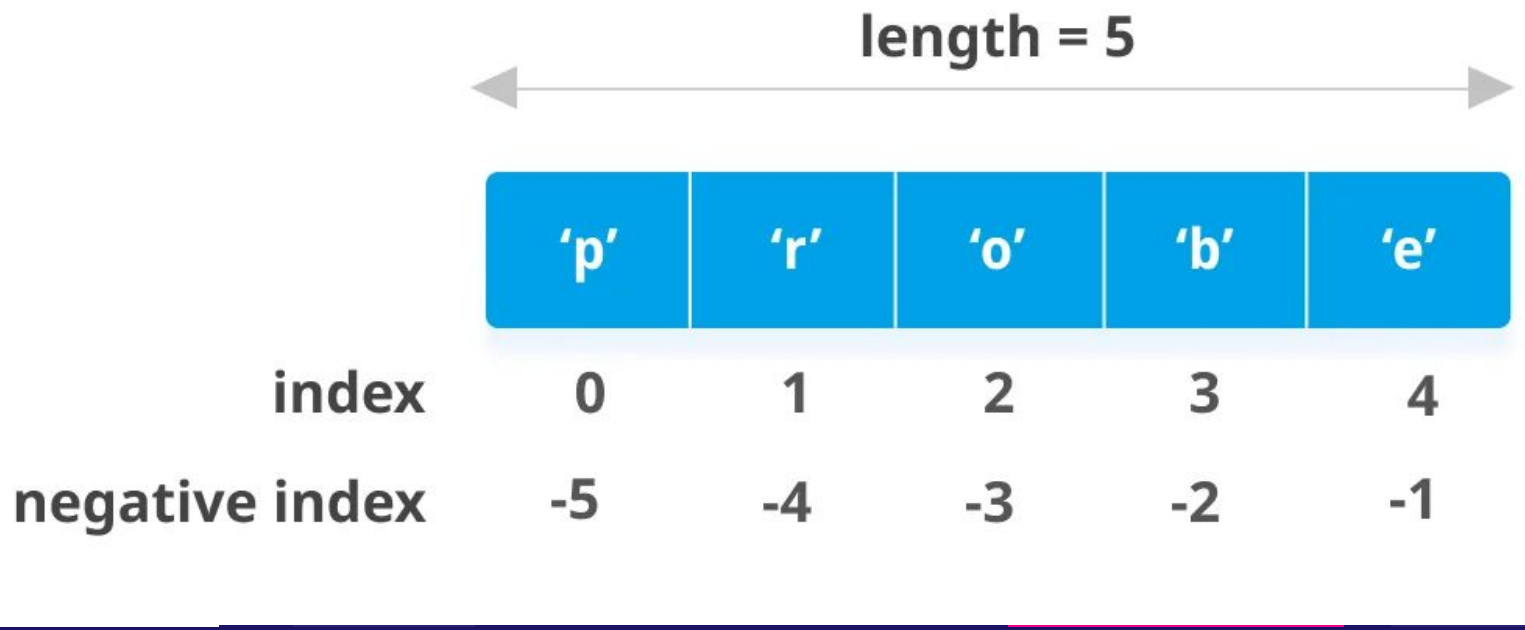
Introduction to programming

Lecture 03

Table of contents

01

LISTS


length = 5
'p'
'r'
'o'
'b'
'e'
index
0
1
2
3
4
negative index
-5
-4
-3
-2
-1

Lists - the basics

```
# Lists are used to store multiple items in a
single variable.
# Lists are created using square brackets -
just like that.

my_list = ["pesho", "gosho", "ivan"]

# Lists can be of any type
list1 = ["pesho", "gosho", "ivan"]
list2 = [1, 2, 3, 4, 5]
list3 = [False, True, False]

# Types can even be mixed
my_list = ["pesho", 34, True, 42, "gosho"]
```

Lists - accessing

```
# List items are ordered, changeable, and
allow duplicate values.
# List items are indexed, the first item has
index [0], the second item has index [1] etc.

my_list = ["pesho", "gosho", "ivan", 42]

first_item = my_list[0]
second_item = my_list[1]
third_item = my_list[2]
# You can count backwards - starting from the
last
last_item = my_list[-1]
```

Lists - manipulating values

```
# List items are changeable
my_list = ["pesho", "gosho", "ivan", 42]

my_list[1] = "ralica"
print(my_list)
# >>> ['pesho', 'ralica', 'ivan', 42]
```

Lists - length

```
# All lists have length - to determine how
many items a list has, use the len() function

my_list = []  # an empty list
print(len(my_list))  # >>> 0

my_list = ["pesho", "gosho", "ivan", 42]
print(len(my_list))  # >>> 4
```

Lists - insert new items

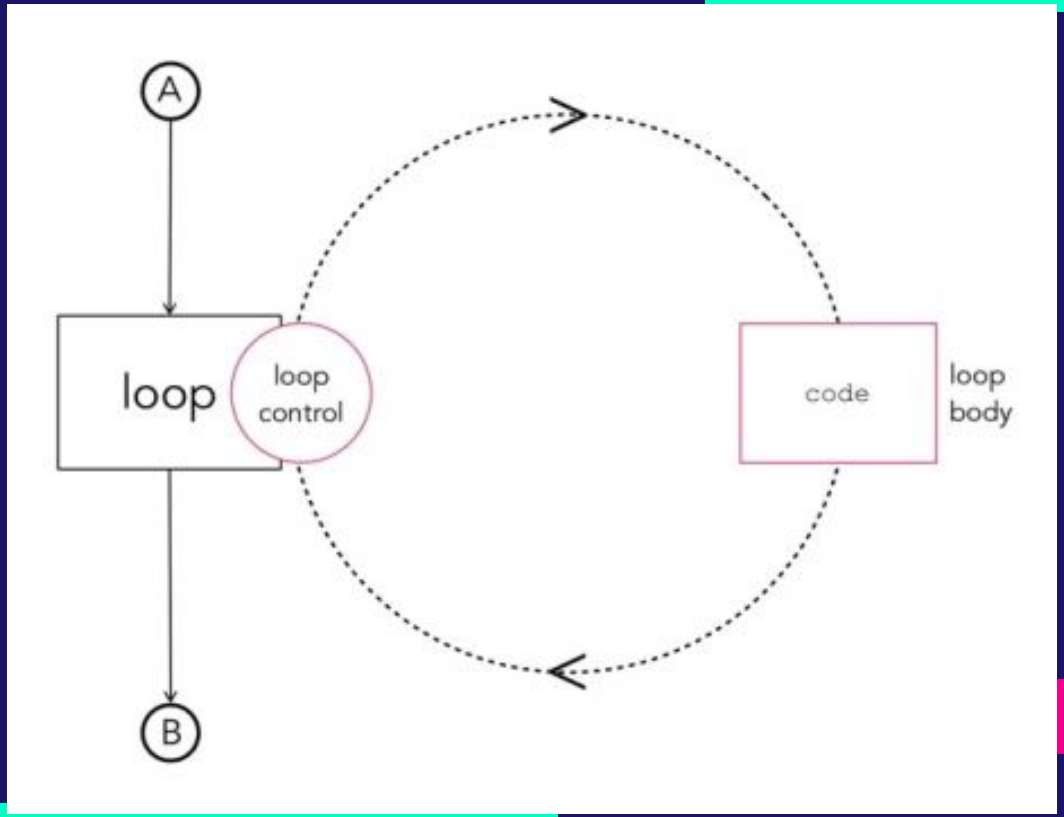
```
# To insert a new list item, without replacing any of the existing values, we can use the insert() method.
# The insert() method inserts an item at the specified index.

my_list = ["pesho", "gosho", "ivan"]
my_list.insert(2, "ralica")
print(my_list)
# >>> ['pesho', 'gosho', 'ralica', 'ivan']

# To add a new list item to the end of the list you can use the append() method.
my_list = ["pesho", "gosho", "ivan"]
my_list.append(10)
print(my_list)
# >>> ['pesho', 'gosho', 'ivan', 10]
```

Lists - remove items

```
# The pop() method removes the specified
index.

my_list = ["pesho", "gosho", "ivan"]
my_list.pop(1)
print(my_list)
# >>> ['pesho', 'ivan']

# The pop() method without an argument will
remove the last item.

my_list = ["pesho", "gosho", "ivan"]
my_list.pop()
print(my_list)
# >>> ['pesho', 'gosho']
```

02

LOOPS


A
loop
loop control
code
loop body
B

Loops - while

```
# With the while loop we can execute a set of statements as long as a condition is True.

i = 1
while i < 6:
    print(i)
    i += 1  # same as `i = i + 1`

# The loop can be 'stuck' in so-called endless loop if the condition is always True.

i = 1
while True:
    print(i)
    i += 1  # same as `i = i + 1`
```

Loops - (break) and continue

```
# With the break statement we can stop the
loop even if the while condition is true.

i = 1
while i < 6:
    if i == 4:
        # when i becomes equal to 4 then loop
will exit
        break
    print(i)
    i += 1  # same as `i = i + 1`
```

Loops - break and (continue)

```
# With the continue statement we can stop the
current iteration, and continue with the next.

i = 1
while i < 6:
    if i % 2:
        # when i is even the loop will
continue with the next iteration
        continue
    print(i)
    i += 1  # same as `i = i + 1`
```

Loops - through lists

```
# We can loop through lists

my_list = ["pesho", "gosho", "ivan"]

i = 0
while i < len(my_list):
    print(my_list[i])
    i += 1  # same as `i = i + 1`
```

Loops - for

```
# A for loop is used for iterating over a sequence (that is either a list, a tuple, a dictionary, a set, or a string).

my_list = ["pesho", "gosho", "ivan"]

for name in my_list:
    print(name)
```

Loops - (break) and continue

```
# With the break statement we can stop the
loop before it has looped through all the
items.

my_list = ["pesho", "gosho", "ivan"]

for name in my_list:
    if name == "gosho":
        break
    print(name)
```

Loops - break and (continue)

```
# With the continue statement we can stop the
current iteration of the loop, andcontinue
with the next.

my_list = ["pesho", "gosho", "ivan"]

for name in my_list:
    if name == "gosho":
        continue
    print(name)
```

Loops - in range()

```
# The range() function returns a sequence of numbers, starting from 0 by default, and increments by 1 (by default), and stops before a specified number.

range(10)
# range(start, stop, step)
# [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]


# To loop through a set of code a specified number of times, we can use the range() function.

for x in range(10):
    print(x)
```


EXERCISE
You can find them in the github repository
https://github.com/genchev99/ia-lectures

THANKS !

CREDITS: This presentation template was created by Slidesgo, incluiding icons by Flaticon, and infographics & images by Freepik.