Programming

Python Data Structures

Vassilis Markos, Mediterranean College

Week 05

Contents

Last Time’s Exercises

In–class Exercise #001

Write a Python function that:

- takes a single integer as an argument, and;
- returns `True` or `False` depending on whether this number is even or odd.

Demonstrate the functionality of your function by properly using it in a simple Python script.

In–class Exercise #002

The Fibonacci numbers, f_n, are a sequence of integer numbers given by the following relation:

$$f_n = f_{n-1} + f_{n-2}, \quad f_0 = 0,\ f_1 = 1.$$

That is, each term is the sum of its previous two. For instance, the first 10 Fibonacci numbers are:

$$0, 1, 1, 2, 3, 5, 8, 13, 21, 34.$$

Write a Python function that takes n as input and prints the n–th Fioinacci number, f_n.

In–class Exercise #003

Write a Python program that:

- asks the user for consecutive positive integers (non–positive input terminates number insertion), and;
- computes and prints their sum and average.

You are required to use **at least three different functions** for your solution and explain your rationale!

In–class Exercise #004

The standard deviation, s, of a set of n numbers $x_1, x_2, \ldots, x_n$ is compute by the following formula:

$$s = \sqrt{\frac{(x_1 - \mu)^2 + (x_2 - \mu)^2 + \cdots + (x_n - \mu)^2}{n}},$$

where μ is the mean value of those numbers.
Write a Python program that asks the user for some non–zero numbers (insertion terminated by inserting 0) and computes their standard deviation. Make sure your program uses **at least two functions!**

In–class Exercise #005

A string is said to be a **palindrome** if it reads the same left–to–right and right–to–left. Write a Python function that:

- takes a single string as an argument, and;
- returns `True` or `False` depending on whether this string is a palindrome.

Demonstrate the functionality of your function by properly using it in a simple Python script.
Hint: In order to access the `i`*–th character of a string named* `s` *you can use the syntax* `s[i]`*.*

In–class Exercise #006

One way to estimate the square root of a positive float, a, is to use the following method:

$$x_n = \frac{1}{2}\left(x_{n-1} + \frac{a}{x_{n-1}}\right),$$

where the first estimate, x_0, is an arbitrary positive float. We say that x_n is an estimation of $\sqrt{a}$ of accuracy $\varepsilon > 0$ if $\|x_n - x_{n-1}\| < \varepsilon$, i.e., if the two latest estimates we have made are no further apart than ε.
Write a Python function that takes x_0, a, and ε as arguments and returns the corresponding estimate, x_n.

In–class Exercise #007

The Towers of the Hanoi is a well–known puzzle where you have to move disks of different sizes one at a time from a peg to another peg with the help of an auxiliary peg and without ever moving a larger disk on top of a smaller one. You can familiarise yourselves with the game below:

`https://www.mathsisfun.com/games/towerofhanoi.html`

Develop a Python function that accepts a positive integer n corresponding to the number of disks on the first peg and prints on screen the required steps to solve the problem.

Python Data Structures

Looking For Numbers

Write a Python script that:

- asks the user for three integers, let a, b, c;
- then asks the user for a fourth integer and checks if it is one of the first three provided before and prints a relevant message on screen

How did you keep track of the first three provided numbers?

A Possible Solution

```python
# source/search_001.py

if __name__ == "__main__":
    a = int(input("Enter a: "))
    b = int(input("Enter b: "))
    c = int(input("Enter c: "))
    d = int(input("Enter a number to look for: "))
    if d == a or d == b or d == c:
        print(True)
    else:
        print(False)
```

Looking For Numbers

Write a Python script that:

- asks the user for as many positive integers as they wish to provide (number insertion stops with the user entering a non–positive number);
- then asks the user for another integer and checks if it is one of the rest ones provided before and prints a relevant message on screen

How did you keep track of the first three provided numbers?

A Possible Solution

```python
# source/search_002.py

def search(needle, haystack):
    for i in range(len(haystack)):
        if needle == haystack[i]:
            return True # This breaks the loop too
    return False

if __name__ == "__main__":
    haystack = []
    n = int(input("Enter a positive integer: "))
    while n > 0:
        haystack.append(n)
        n = int(input("Enter a positive integer: "))
    needle = int(input("Enter a number to look for: "))
    print(search(needle, haystack))
```

Python Lists

In the above solution we used on of Python’s best–selling data structures: a **list:**

- Lists can be thought off as a collection of items in a single place.
- The above is not accurate in terms of memory allocation, but it is a very good mental model for Python’s lists.
- We can initialise an empty list as in line 10, by using `[]`.
- We can add elements to a list by using `.append()`.
- We can get the length of a list by using `len()`.
- We can access the `i`–th element of list `a` (0 indexed) by `a[i]`.

Heads up!

Indexing

- Suppose you have a few apples in front of you. How would you count them?

Indexing

- Suppose you have a few apples in front of you. How would you count them?
    - 1, 2, 3,...

Indexing

- Suppose you have a few apples in front of you. How would you count them?
  - 1, 2, 3,...
- When it comes to lists and indexing in general, things do not work that way for most programming languages, Python included.

Indexing

- Suppose you have a few apples in front of you. How would you count them?
    - 1, 2, 3,...
- When it comes to lists and indexing in general, things do not work that way for most programming languages, Python included.
- For instance, the elements of a list like `a=[3, 6, 2, 5]` are indexed as follows:

$$a = [\underset{0}{3}, \underset{1}{6}, \underset{2}{2}, \underset{3}{5}]$$

- So, the first element is at position 0, the second at position 1, etc.

Indexing

- Suppose you have a few apples in front of you. How would you count them?
  - 1, 2, 3,...
- When it comes to lists and indexing in general, things do not work that way for most programming languages, Python included.
- For instance, the elements of a list like `a=[3, 6, 2, 5]` are indexed as follows:

$$a = [\underset{0}{3}, \underset{1}{6}, \underset{2}{2}, \underset{3}{5}]$$

- So, the first element is at position 0, the second at position 1, etc.
- So, the last element is at position...?

Indexing

- Suppose you have a few apples in front of you. How would you count them?
  - 1, 2, 3,...
- When it comes to lists and indexing in general, things do not work that way for most programming languages, Python included.
- For instance, the elements of a list like `a=[3, 6, 2, 5]` are indexed as follows:

$$a = [\underset{0}{3}, \underset{1}{6}, \underset{2}{2}, \underset{3}{5}]$$

- So, the first element is at position 0, the second at position 1, etc.
- So, the last element is at position...?
  - `len(a) - 1`. Be careful about that!

Negative Indexing

What do you expect the following to print?

```python
# source/indexing_001.py

if __name__ == "__main__":
    a = [5, 6, 3, 1]
    print(a[-1], a[-2])
```

Negative Indexing

What do you expect the following to print?

```python
# source/indexing_001.py

if __name__ == "__main__":
    a = [5, 6, 3, 1]
    print(a[-1], a[-2])
```

- Python allows negative indexing, interpreted as follows:

```
a[-n] = a[len(a) - n]
```

Negative Indexing

What do you expect the following to print?

```python
# source/indexing_001.py

if __name__ == "__main__":
    a = [5, 6, 3, 1]
    print(a[-1], a[-2])
```

- Python allows negative indexing, interpreted as follows:

  `a[-n] = a[len(a) - n]`

- So, **negative indices** count starting from the **end of the list!**

Looping Over A List

```python
# source/search_003.py

def search(needle, haystack):
    for x in haystack:
        if needle == x:
            return True # This breaks the loop too
    return False

if __name__ == "__main__":
    haystack = []
    n = int(input("Enter a positive integer: "))
    while n > 0:
        haystack.append(n)
        n = int(input("Enter a positive integer: "))
    needle = int(input("Enter a number to look for: "))
    print(search(needle, haystack))
```

List Membership

We can check list membership directly, using the `in` keyword:

```python
# source/search_004.py

if __name__ == "__main__":
    haystack = []
    n = int(input("Enter a positive integer: "))
    while n > 0:
        haystack.append(n)
        n = int(input("Enter a positive integer: "))
    needle = int(input("Enter a number to look for: "))
    if needle in haystack:
        print(True)
    else:
        print(False)
```

Popping From A List

What do you expect the following to print?

```python
# source/operations_001.py

if __name__ == "__main__":
    a = [5, 2, 9, 0, 4]
    for _ in range(len(a)):
        x = a.pop()
        print(x)
```

Popping From A List

What do you expect the following to print?

```python
# source/operations_001.py

if __name__ == "__main__":
    a = [5, 2, 9, 0, 4]
    for _ in range(len(a)):
        x = a.pop()
        print(x)
```

- It prints the elements of the list in reverse order.

Popping From A List

What do you expect the following to print?

```python
# source/operations_001.py

if __name__ == "__main__":
    a = [5, 2, 9, 0, 4]
    for _ in range(len(a)):
        x = a.pop()
        print(x)
```

- It prints the elements of the list in reverse order.
- So, what does `.pop()` do?

Popping From A List

- `.pop()` can actually pop any element from a Python list and not just the last one.
- To achieve that, just call it using the corresponding index.
    - For instance, to pop the first element of a list one should write...

Popping From A List

- `.pop()` can actually pop any element from a Python list and not just the last one.
- To achieve that, just call it using the corresponding index.
    - For instance, to pop the first element of a list one should write...
    - `.pop(0)`.
    - For instance, to pop the third element of a list one should write...

Popping From A List

- `.pop()` can actually pop any element from a Python list and not just the last one.
- To achieve that, just call it using the corresponding index.
  - For instance, to pop the first element of a list one should write...
  - `.pop(0)`.
  - For instance, to pop the third element of a list one should write...
  - `.pop(2)`.
  - For instance, to pop the 25th element of a list one should write...

Popping From A List

- `.pop()` can actually pop any element from a Python list and not just the last one.
- To achieve that, just call it using the corresponding index.
    - For instance, to pop the first element of a list one should write...
    - `.pop(0)`.
    - For instance, to pop the third element of a list one should write...
    - `.pop(2)`.
    - For instance, to pop the 25th element of a list one should write...
    - `.pop(24)`.
- `.pop()` throws an index error if asked to pop an element out of range.

Popping Stuff Costs...

The following script times `.pop()` performance when popping from various places in a list. What do you expect it to print?

```python
# source/operations_002.py
from timeit import timeit # To time stuff

if __name__ == "__main__":
    N = 100000
    a = [x for x in range(N)] # a = [0,1,...,N-1]
    b = [x for x in range(N)] # b = [0,1,...,N-1]
    c = [x for x in range(N)] # c = [0,1,...,N-1]
    pop_last = timeit(lambda : a.pop(), number = N)
    pop_first = timeit(lambda : b.pop(0), number = N)
    pop_middle = timeit(lambda : c.pop(len(c) // 2), number = N)
    print(f"last:   {pop_last}\nfirst:  {pop_first}\nmiddle: {pop_middle}")
```

Popping Costs

A typical output might look like the following:

```
last:   0.004077139998116763
first:  0.8199967760010622
middle: 0.347551475999353
```

- Why do you think is that?
- Roughly, because `.pop()` has to push all elements on the right of the popped element one position left, which increases as we approach the list start.

.insert()

What do you expect this to print?

```python
# source/operations_003.py
from timeit import timeit # To time stuff

if __name__ == "__main__":
    N = 100000
    a = []
    b = []
    c = []
    lst_append = timeit(lambda : a.append(4), number = N)
    insert_end = timeit(lambda : b.insert(-1, 4), number = N)
    insert_start = timeit(lambda : c.insert(0, 4), number = N)
    print(f"append:       {lst_append}\ninsert end:   {insert_end}\ninsert
     start: {insert_start}")
```

`.insert()`

Python’s `.insert()` allows insert elements at an arbitrary position in a list.

- Inserting at the end of the list, costs roughly as for `.append()`.

`.insert()`

Python's `.insert()` allows insert elements at an arbitrary position in a list.

- Inserting at the end of the list, costs roughly as for `.append()`.
- Not the same, though, so, if you always insert at the end, prefer `.append()`.

`.insert()`

Python's `.insert()` allows insert elements at an arbitrary position in a list.

- Inserting at the end of the list, costs roughly as for `.append()`.
- Not the same, though, so, if you always insert at the end, prefer `.append()`.
- Why does it cost that more to insert in the beginning of the list?

`.insert()`

Python's `.insert()` allows insert elements at an arbitrary position in a list.

- Inserting at the end of the list, costs roughly as for `.append()`.
- Not the same, though, so, if you always insert at the end, prefer `.append()`.
- Why does it cost that more to insert in the beginning of the list?
- Because, as with `.pop()`, we have to push all subsequent elements one position to the right!

`.insert()`

Python's `.insert()` allows insert elements at an arbitrary position in a list.

- Inserting at the end of the list, costs roughly as for `.append()`.
- Not the same, though, so, if you always insert at the end, prefer `.append()`.
- Why does it cost that more to insert in the beginning of the list?
- Because, as with `.pop()`, we have to push all subsequent elements one position to the right!
- So, design your algorithms to avoid inserting that much to the beginning of a list!

Not relevant for the assessment of the module!

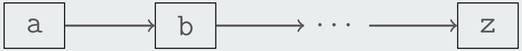
Lists In A Nutshell

A simply linked list looks much like what follows in memory:

$$\begin{tikzcd} \boxed{\texttt{a}} \arrow[r] & \boxed{\texttt{b}} \arrow[r] & \cdots \arrow[r] & \boxed{\texttt{z}} \end{tikzcd}$$

Lists In A Nutshell

A simply linked list looks much like what follows in memory:



- Each node points to its next one (there are some peculiarities regarding the first and last nodes but ignore them for now).

Not relevant for the assessment of the module!

Lists In A Nutshell

A simply linked list looks much like what follows in memory:

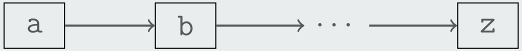
$$\begin{tikzcd}
\boxed{\texttt{a}} \arrow[r] & \boxed{\texttt{b}} \arrow[r] & \cdots \arrow[r] & \boxed{\texttt{z}}
\end{tikzcd}$$

- Each node points to its next one (there are some peculiarities regarding the first and last nodes but ignore them for now).
- So, how much time does it cost to find the last node?

Not relevant for the assessment of the module!

Lists In A Nutshell

A simply linked list looks much like what follows in memory:

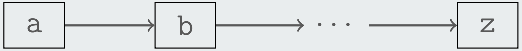


- Each node points to its next one (there are some peculiarities regarding the first and last nodes but ignore them for now).
- So, how much time does it cost to find the last node?
  - It depends on the length of the list: the lengthier the list, the longer it takes to get to the last element.

Not relevant for the assessment of the module!

Lists In A Nutshell

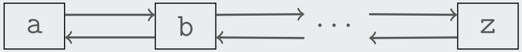
A simply linked list looks much like what follows in memory:



- Each node points to its next one (there are some peculiarities regarding the first and last nodes but ignore them for now).
- So, how much time does it cost to find the last node?
    - It depends on the length of the list: the lengthier the list, the longer it takes to get to the last element.
- Any ideas to make navigation easier?

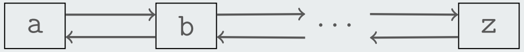
Lists In A Nutshell

A **doubly** linked list looks much like what follows in memory:


$$\begin{tikzcd} a \arrow[r, shift left] & b \arrow[l, shift left] \arrow[r, shift left] & \cdots \arrow[l, shift left] \arrow[r, shift left] & z \arrow[l, shift left] \end{tikzcd}$$

Lists In A Nutshell

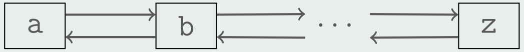
A **doubly** linked list looks much like what follows in memory:



- Here, each node points to both its next and previous ones (again, first and last nodes are treated separately).

Lists In A Nutshell

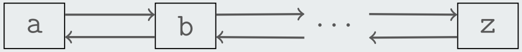
A **doubly** linked list looks much like what follows in memory:



- Here, each node points to both its next and previous ones (again, first and last nodes are treated separately).
- So, how much time does it cost to find the last node?

Lists In A Nutshell

A **doubly** linked list looks much like what follows in memory:



- Here, each node points to both its next and previous ones (again, first and last nodes are treated separately).
- So, how much time does it cost to find the last node?
    - Roughly, half as before.

Python Lists Are Not Lists!

Even if named as lists, Python lists are actually (dynamically sized) arrays:

- That is, all elements are stored in consecutive memory locations.
- Whenever the list gets (almost) full, it expands to accommodate more elements, if possible.
- Whenever the list gets too empty, it shrinks to free memory.
- So, accessing an element is actually faster than for a list, it just comes at the cost of occasionally resizing the list.

Python Tuples: Immutable Lists

What does the following print?

```python
# source/tuples_001.py

if __name__ == "__main__":
    x = (2, 3, 4)
    print(x)
    print(x[1])
    x[0] = 4
    print(x)
```

Python Tuples: Immutable Lists

Probably, you get something like the following:

```
(2, 3, 4)
3
Traceback (most recent call last):
File "tuples_001.py", line 7, in <module>
x[0] = 4
TypeError: 'tuple' object does not support item assignment
```

Python Tuples: Immutable Lists

Probably, you get something like the following:

```
(2, 3, 4)
3
Traceback (most recent call last):
File "tuples_001.py", line 7, in <module>
x[0] = 4
TypeError: 'tuple' object does not support item assignment
```

- Python tuples are just like lists, but we cannot change any of their attributes, i.e.:
  - their contents;
  - their length;
- So we can just treat them as, roughly, constant (immutable) lists.

Names And Ages

Write a Python script that:

- asks the user to provide as many times as they want to pairs of the form `("name", age)`, which correspond to the user's friends names and ages (data insertion terminates when an empty name is provided);
- then asks the use to enter a certain name, and;
- prints the age of the person on screen if the name has been previously entered, or prints an error message otherwise.

A Solution Using Lists And Tuples

```python
# source/tuples_002.py
if __name__ == "__main__":
    people = []
    name = input("Enter a name: ")
    while name != "":
        age = int(input(f"Enter {name}'s age: "))
        people.append((name, age)) # Append a tuple to our list
        name = input("Enter a name: ")
    needle = input("Enter a name to look for: ")
    found = False
    for name, age in people:
        if needle == name:
            print(f"{needle}'s age is: {age}")
            found = True
            break
    if not found:
        print(f"{needle} not found!")
```

A Solution Using Python Dictionaries

```python
# source/dictionaries_001.py

if __name__ == "__main__":
    people = dict() # An empty dictionary
    name = input("Enter a name: ")
    while name != "":
        age = int(input(f"Enter {name}'s age: "))
        people[name] = age # Map 'name' to 'age'
        name = input("Enter a name: ")
    needle = input("Enter a name to look for: ")
    found = False
    if needle in people.keys():
        print(f"{needle}'s age is: {people[needle]}")
    else:
        print(f"{needle} not found!")
```

Python Dictionaries

Dictionaries are like mathematical functions, in the sense that they map keys to a **single value**, e.g.:

```
dict = {
    key_1: value_1,
    key_2: value_2,
    ...
    key_n: value_n
}
```

We can initialise a dictionary by directly providing its values in the above form.

Dictionary Utililities

- `.keys()` returns a Python iterable (not a list!) containing the dictionary's keys.
- `.values()` returns a Python iterable (not a list!) containing the dictionary's values.
- `.items()` returns a Python iterable (not a list!) containing the dictionary's keys and values as a tuple of 2.
- For all the above, we can use typical list–like operations, but they above are not lists! If we want to use them as lists (not recommended) we can wrap them using `list()`.

Are Dictionaries Better?

```python
# source/dictionaries_002.py
from timeit import timeit
from random import randint

def list_search(l, key):
    for l_key, value in l:
        if l_key == key:
            return value
    return -1

if __name__ == "__main__":
    L = 0; U = 10000; N = U - L + 1
    a = { x: randint(L, U) for x in range(N) }
    b = [ (x, randint(L, U)) for x in range(N) ]
    dict_time = timeit(lambda : a[randint(L, U)], number = U - L)
    list_time = timeit(lambda : list_search(b, randint(L, U)), number = U - L)
    print(f"dict: {dict_time}\nlist: {list_time}")
```

Are Dictionaries Better?

- As a rough idea, `a` is a dictionary, while `b` is a similar structure, represented as a list of tuples.

Are Dictionaries Better?

- As a rough idea, `a` is a dictionary, while `b` is a similar structure, represented as a list of tuples.
- Then, we time value retrieval based on random keys. A typical output should look like:

```
dict: 0.0038980588996128384
list: 0.8523417840042384
```

Are Dictionaries Better?

- As a rough idea, `a` is a dictionary, while `b` is a similar structure, represented as a list of tuples.
- Then, we time value retrieval based on random keys. A typical output should look like:

```
dict: 0.003898058996128384
list: 0.8523417840042384
```

- This is a chaotic difference! Why is that?

Are Dictionaries Better?

- As a rough idea, `a` is a dictionary, while `b` is a similar structure, represented as a list of tuples.
- Then, we time value retrieval based on random keys. A typical output should look like:

```
dict: 0.0038980589961283 84
list: 0.8523417840042384
```

- This is a chaotic difference! Why is that?
- You will learn more in your Data Structures module, so stay tuned...

Fun Time!

In–class Exercise #001

Using appropriate Python data structures write a Python script that:

- Asks the user to provide student names along their grade at the Programming module.
- Stores the above in an appropriate data structure.
- Asks the user to provide a student name and prints their grade.
- The script should terminate when the user enters an empty student name.

In–class Exercise #002

Using appropriate Python data structures write a Python script that:

- asks the user to provide positive integers (number insertion is terminated when the user inserts a non–positive integer);
- computes the sum of each integer's proper divisors;
- then asks the user repeatedly for numbers and prints on screen the sum of its proper divisors;
- the script should terminate when the user asks for the divisor sum of a non–positive integer.

In–class Exercise #003

Read the following Wikipedia lemma on the Sieve of Eratosthenes:

`https://en.wikipedia.org/wiki/Sieve_of_Eratosthenes`

Then, using appropriate Python data structures, implement the sieve algorithm that computes all prime numbers up to a certain provided positive integer, `n`.

In–class Exercise #004

Since programming is not only about reading simple input from the user and returning a nicely computed output, but also about crafting some more complex projects, how about we build our own game?
In today's materials, under the following link:

`../labs/Programming_Lab_01.pdf`

you can find the first part of a Lab series for this course. Follow the instructions found therein to start working on it!

In–class Exercise #005

Start working on all Labs found in today's materials `homework` directory. To help me asses those files, you can name them as follows:

```
task_xxx.py
```

where `xxx` is the number of the task. For instance, task 5 file could be named `task_005.py`.
Submit your work via email at: `v.markos@mc-class.gr`

Homework

- In this week's materials, under the `homework` directory, you can find some Python programming Tasks. Complete as many of them as you can (preferably all).
- This is important, since tasks such as those provided with this lecture's materials will most probably be part of your course assessment portfolio. So, take care to solve as many of those tasks as possible!
- Share your work at: `v.markos@mc-class.gr`

Any Questions?

Do not forget to fill in the questionnaire shown right!

https://forms.gle/dKSrmE1VRVWqxBGZA