

# Algorithmic and advanced Programming in Python

Remy Belmonte [remy.belmonte@dauphine.eu](mailto:remy.belmonte@dauphine.eu)

Lab 1

# Outline

1. Some algorithmic complexity question
2. Introduction to python anywhere
3. Do algorithm for double linked list

# Some complexity and running time questions 1/2

**Problem-21** Find the complexity of the below recurrence:

$$T(n) = \begin{cases} 3T(n-1), & \text{if } n > 0, \\ 1, & \text{otherwise} \end{cases}$$

**Problem-22** Find the complexity of the below recurrence:

$$T(n) = \begin{cases} 2T(n-1) - 1, & \text{if } n > 0, \\ 1, & \text{otherwise} \end{cases}$$

**Problem-23** What is the running time of the following function?

```
def Function(n):  
    i = s = 1  
    while s < n:  
        i = i+1  
        s = s+i  
        print("s")
```

Function(20)

# Some complexity and running time questions 2/2

**Problem-24** Find the complexity of the function given below.

```
def Function(n):
    i = 1
    count = 0
    while i*i < n:
        count = count + 1
        i = i + 1
    print(count)
Function(20)
```

**Problem-25** What is the complexity of the program given below:

```
def Function(n):
    count = 0
    for i in range(n/2, n):
        j = 1
        while j + n/2 <= n:
            k = 1
            while k <= n:
                count = count + 1
                k = k * 2
            j = j + 1
    print (count)
Function(20)
```

# Python anywhere

- We assume that you've got a little bit of basic Python and HTML knowledge – for example, that you've done an online course in both of them. Everything else we'll explain as we go along. Let's get started!

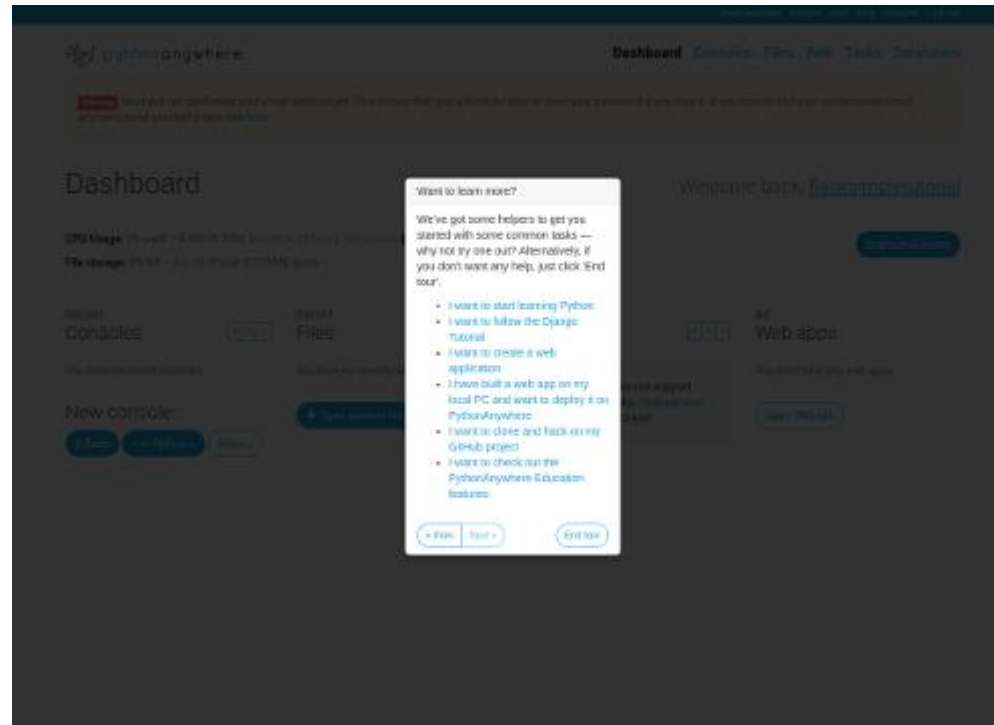
# First steps

- Firstly, [create a PythonAnywhere account](#) if you haven't already. A free “Beginner” account is enough for this tutorial.
- Once you've signed up, you'll be taken to the dashboard, with a tour window. It's worth going through the tour so that you can learn how the site works – it'll only take a minute or so.

# Tour

# Tour

- At the end of the tour you'll be presented with some options to “learn more”. You can just click “End tour” here, because this tutorial will tell you all you need to know.





# Confirm your email address

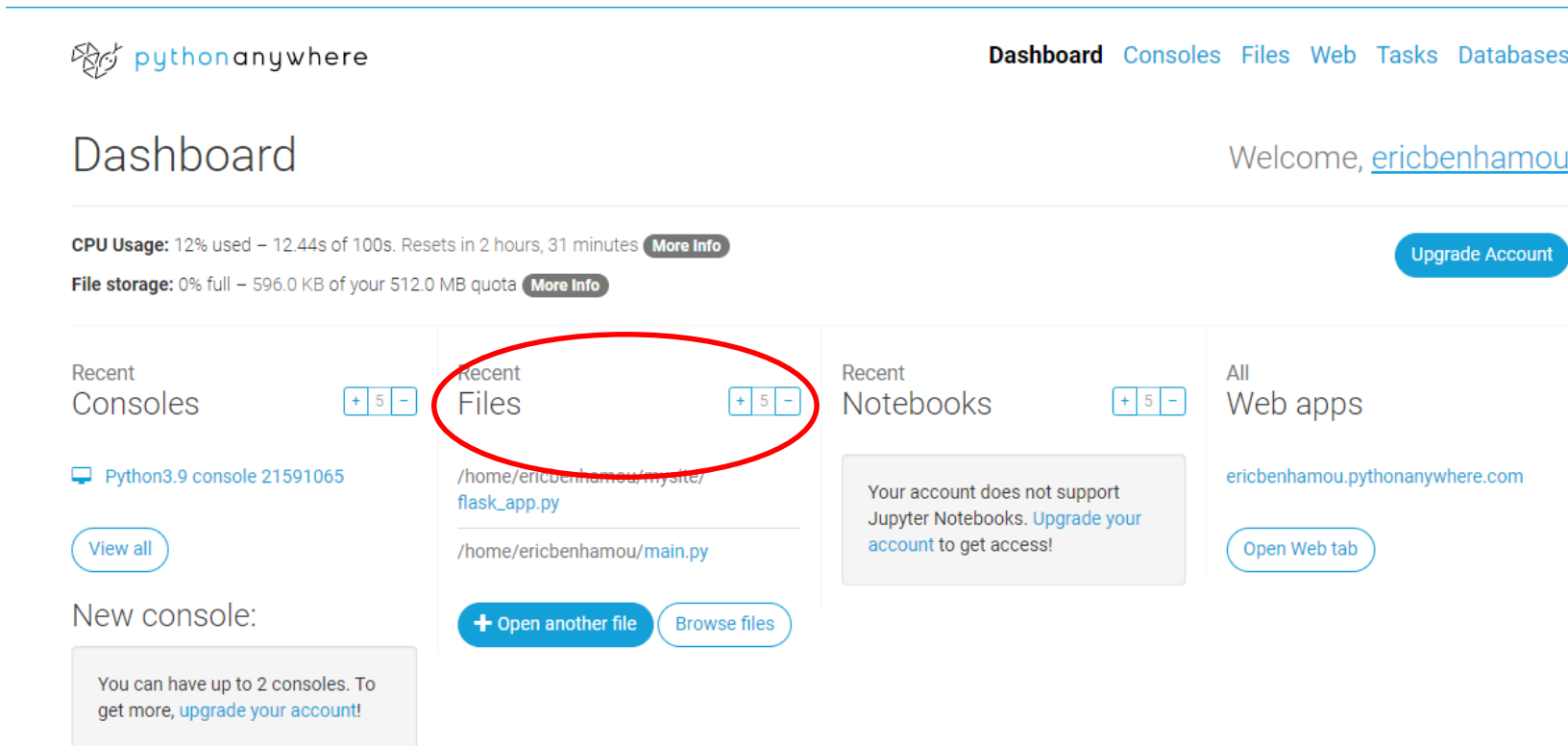
- Now you're presented with the PythonAnywhere dashboard. I recommend you check your email and confirm your email address – otherwise if you forget your password later, you won't be able to reset it.
-

# Dashboard

The screenshot shows the PythonAnywhere dashboard interface. At the top, there is a navigation bar with links for 'Send Feedback', 'Forum', 'Help', 'Blog', 'Account', and 'Logout'. The main header includes the PythonAnywhere logo and a navigation menu with 'Dashboard', 'Consoles', 'Files', 'Web', 'Tasks', and 'Databases'. A yellow warning banner states: 'Warning: You have not confirmed your email address yet. This means that you will not be able to reset your password if you lose it. If you cannot find your confirmation email address, send yourself a new one [here](#).' Below this, the dashboard title 'Dashboard' is displayed alongside a welcome message: 'Welcome back, [\[flasksimpletutorial\]](#)'. The dashboard provides system metrics: 'CPU Usage: 0% used - 0.00s of 100s. Resets in 23 hours, 59 minutes' (with a 'More info' link) and 'File storage: 0% full - 0.0 KB of your 512.0 MB quota'. There are four main sections: 'Recent Consoles' (with a 'New console:' section below it offering '1 Bash', '1 Python', and 'More...' buttons), 'Recent Files' (with 'Open another file' and 'Browse files' buttons), 'Recent Notebooks' (with a message: 'Your account does not support Jupyter Notebooks. Upgrade your account to get access!'), and 'All Web apps' (with an 'Open Web lab' button). An 'Upgrade Account' button is located in the top right corner of the dashboard area.

# Create a first python file to play with list

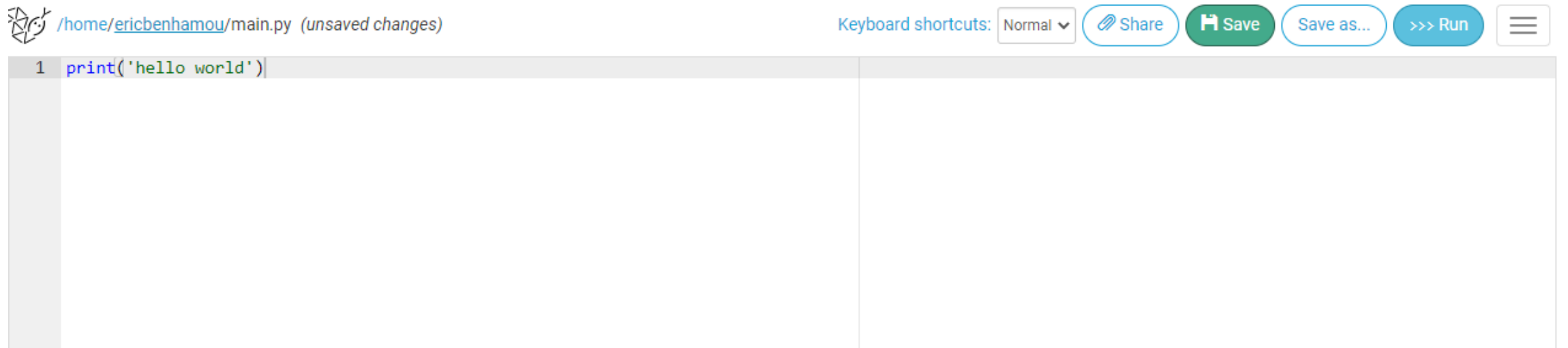
- Now, click on the “Files” link near the center to create your first file on the server



The screenshot shows the PythonAnywhere dashboard. At the top, there is a navigation bar with links for Dashboard, Consoles, Files, Web, Tasks, and Databases. The 'Files' link is circled in red. Below the navigation bar, the dashboard displays the user's name 'ericbenhamou' and a 'Welcome' message. There are two status bars: 'CPU Usage: 12% used - 12.44s of 100s. Resets in 2 hours, 31 minutes' and 'File storage: 0% full - 596.0 KB of your 512.0 MB quota'. Below these are four main sections: 'Recent Consoles', 'Recent Files', 'Recent Notebooks', and 'All Web apps'. The 'Recent Files' section is highlighted with a red circle and shows two files: '/home/ericbenhamou/mysite/flask\_app.py' and '/home/ericbenhamou/main.py'. There are buttons for '+ Open another file' and 'Browse files' at the bottom of the 'Recent Files' section. A message in the 'Recent Notebooks' section states: 'Your account does not support Jupyter Notebooks. Upgrade your account to get access!'. A 'View all' button is present under 'Recent Consoles', and an 'Open Web tab' button is present under 'All Web apps'. A 'New console:' section at the bottom left contains a message: 'You can have up to 2 consoles. To get more, upgrade your account!'.

# Do your first step in python interpreter

- Type `print('hello world')`
- Click on `>>>>Run`



The screenshot shows a web-based Python IDE interface. At the top left, there is a logo and the file path `/home/ericbenhamou/main.py` with the text `(unsaved changes)`. To the right of the path is a 'Keyboard shortcuts' dropdown menu set to 'Normal'. Further right are four buttons: 'Share' (with a link icon), 'Save' (with a floppy disk icon), 'Save as...' (with a document icon), and '>>> Run' (with a play icon). Below the toolbar is a code editor area with a single line of code: `1 print('hello world')`. The editor has a light gray background and a vertical line indicating the current cursor position at the end of the line.

# You should get this!

```
hello world  
>>> █
```

# Do not forget to set your professor to me

pythonanywhere Dashboard Consoles Files Web Tasks Databases

In the private browsing tab, sign up for a second PythonAnywhere account. You can use the same email address as your main account if you like. ×

Once the account has been created, click through to its **Account** page, and find the **Teacher** tab. Enter your main account's username into this field, and hit enter.

← → Close this tutorial

Upgrade/Downgrade Account Security Email Education API Token System Image

As part of our [Education beta](#), users can now nominate another user to be their "teacher". [Find out more.](#)

## Your teacher

You can set who your teacher is below. **Warning!** this means they have [full access](#) to all your consoles, files and folders on PythonAnywhere, so you should make sure it's someone you trust!

You can revoke their access at any time by resetting the input field below. Blank means you don't have a teacher.

[Enter your teacher's username](#)

## Your students

These are people who have named you as their teacher. You have [full access](#) to all your students' consoles, files and folders on PythonAnywhere.

You can remove them as students at any time using the list below.

*You have no students.*

# Set it to remybelmonte

[Send feedback](#) [Forums](#) [Help](#) [Blog](#) [Account](#) [Log out](#)



[Dashboard](#) [Consoles](#) [Files](#) [Web](#) [Tasks](#) [Databases](#)

[Upgrade/Downgrade Account](#)

[Security](#)

[Email](#)

[Education](#)

[API Token](#)

[System Image](#)

As part of our [Education beta](#), users can now nominate another user to be their "teacher". [Find out more.](#)

## Your teacher

You can set who your teacher is below. **Warning!** this means they have full access to all your consoles, files and folders on PythonAnywhere, so you should make sure it's someone you trust!

You can revoke their access at any time by resetting the input field below. Blank means you don't have a teacher.

[remybelmonte](#)

## Your students

These are people who have named you as their teacher. You have full access to all your students' consoles, files and folders on PythonAnywhere.

You can remove them as students at any time using the list below.

*You have no students.*

# Now play with linked list


- Download the file

[Advanced Programming & Algo - 1 - Lab resource.py](#)

in moodle.

---

## Evaluation mechanism

 Final note = 0.3 Project + 0.7 Exam


---

## Master class 1

 [Advanced Programming & Algo - 1 - MasterClass](#)

 [Advanced Programming & Algo - 1 - Lab](#)

 [Advanced Programming & Algo - 1 - Lab Resource.py](#)

 [Advanced Programming & Algo - 1 - Lab - Solutions](#)



# Now play with linked list

- The file [Advanced Programming & Algo - 1 - Lab resource.py](#) in moodle contains an incomplete implementation of a Python LinkedList class. Take a minute to look over this code. Open a Python interpreter and experiment with creating a LinkedList object and calling the methods that have already been implemented.

# Exercise: question 1

1. Implement the count method, which should return a count of the number of times that the given item is found in the list.

# Question 2: Index method

- Implement the index method. This will be very similar to the included `__contains__` method, except that it needs to return the index of the element if it is found, rather than a simple boolean. Thus, you will need to track the current index as you traverse the linked list.

# Question 3

- Implement the append method, which should add a new element onto the tail of the list. You must also remember to handle the special case when the list is empty. Given the current implementation, there is no  $O(1)$  way to add an element to the tail of the list. You have two options to implement this function:
- Iterate to the end of the list, finding the last node and adding the new node after that node. This will be  $O(n)$  but that is ok for the purposes of this lab.
- Add a `_tail` reference to the `LinkedList` class and use it to add a new item in  $O(1)$  time. This is a better solution, but will require you to change several other functions to properly maintain the tail pointer.

# Question 4: equal and not equal

- Implement the `__eq__` and `__ne__` methods. For these functions, equality should be defined as follows: both lists have the same number of elements, and each pair of corresponding elements in the list are also equal (as defined by the `==` operator). You should implement only one of these operators from scratch; the other should delegate to the first.

# Optional 1

- (Optional) Implement the insert method. One way of organizing this method is to work through the following steps:
- If necessary, convert the index value from negative to positive.
- Raise an exception if the index is out of bounds.
- Create a new node to contain the item being inserted.
- Check to see if insertion should occur at the head, if so handle that as a special case.
- If the item is not being inserted at the head, use a loop to step through the correct number of nodes, keeping track of a `prev_node` and a `cur_node`. Insert the new node at the appropriate location.
- Increment the size of the list.

# Optional 2

- (Optional) Implement the remove method. You will need to traverse the list looking for the item to remove, keeping track of the predecessor node so that you can reconnect the two separate parts of the list after removal. If the item is not found in the list, the method should raise a ValueError.