

Week 6 Tutorial

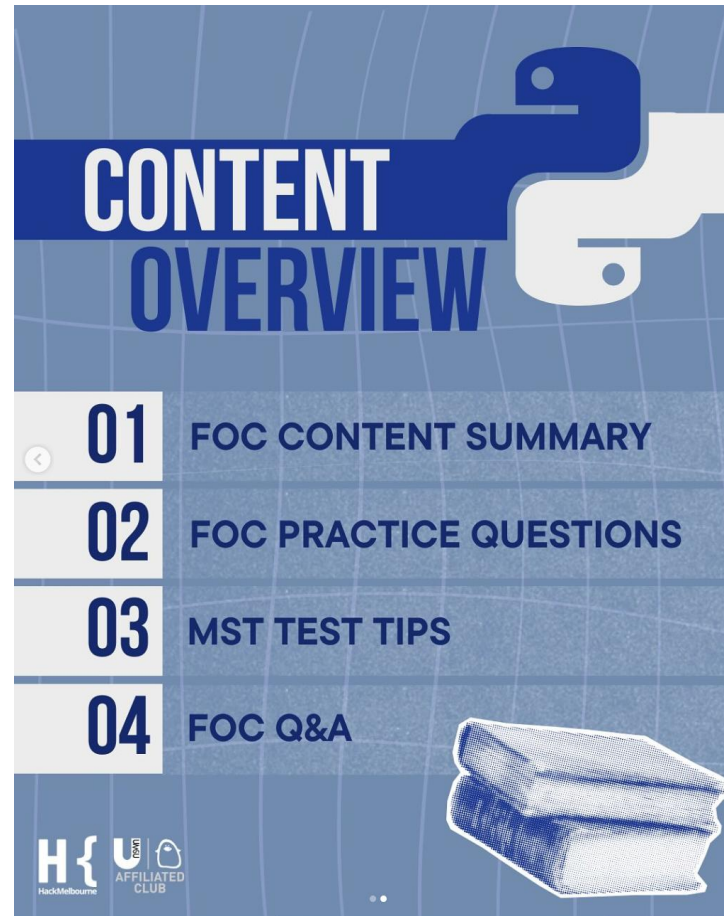
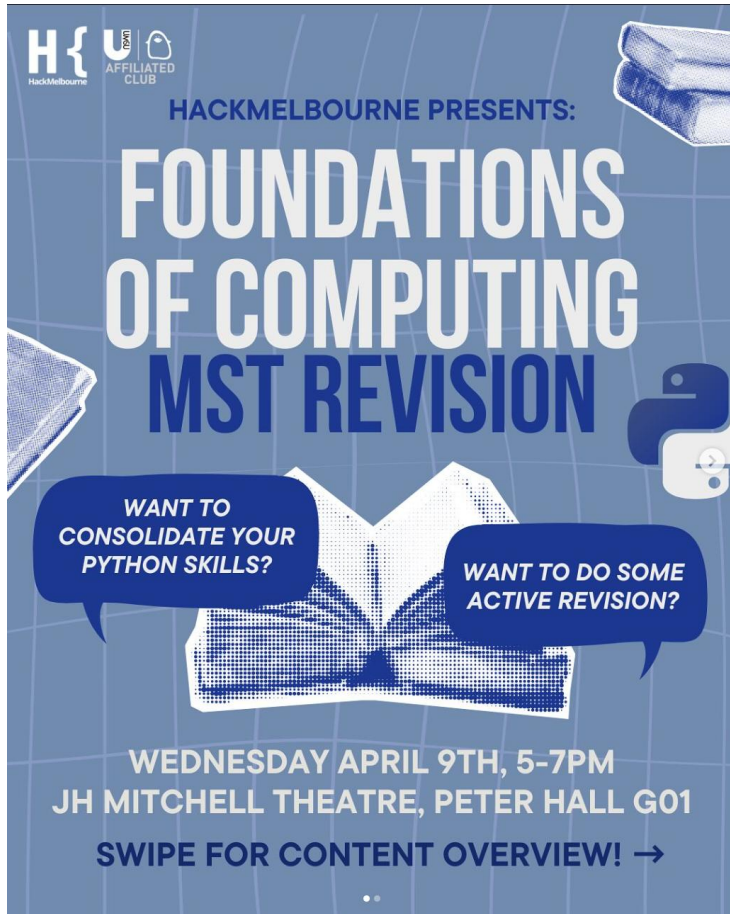
COMP10001 – Foundations of Computing

Semester 1, 2025

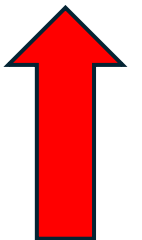
Clement Chau

- Dictionaries, Sets, and Containers
- Global and Local Namespaces (Scope)

Before we start,



Scan Me



Before we start (2),

Useful Tips for Your Ed Project/Worksheets

- **Most important:** Start Early!
- To fix the indentation of multiple lines, you can highlight a block of code and:
 - indent them by hitting *tab*
 - remove indentation by hitting *shift + tab*
- If your line is too long (hopefully not the case), use brackets (suggested by PEP8 guidelines) or backslash `\` to divide this line into two.

Example: The below snippets of code below are equivalent.

```
if year == 2024 and month == "Aug" and day == 26 and lecturer in ("SC", "GB", "KV"):
    print("Aha it's COMP10001")
```

```
if (year == 2024 and month == "Aug" and day == 26 and
    lecturer in ("SC", "GB", "KV")):
    print("Aha it's COMP10001")
```

- Use the terminal to test your functions with different arguments. You can hit `↑` on your keyboard to get the previous command that you entered. Useful!
- You can hit “mark” as many times as you want, and we will mark your last submission before the deadline.

Question 1, assume that `seq` is a list. Write one line of code to:

1. Sort `seq` in-place (i.e. mutate the list)

➡ `seq.sort()`

2. Assign the sorted version of `seq` to a new variable `new_seq`. Do not change the original list.

➡ `new_seq = sorted(seq)`

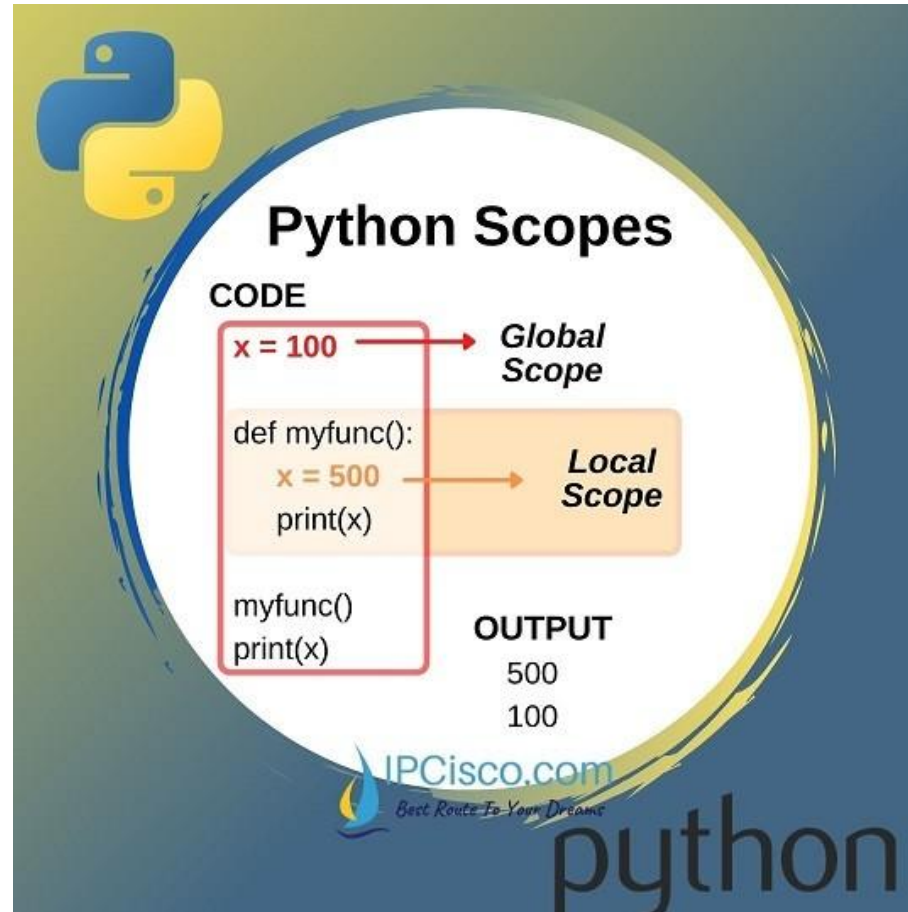
3. Add the string `"hi"` to the end of `seq`.

➡ `seq.append("hi")`

OR

`seq = seq + ["hi"]`

Revision: Python Scopes!



Question 2, what is the output of this code?

2. What is the output of the following code? Classify the variables by which namespace they belong in.

```
def foo(x, y):  
    a = 42  
    x, y = y, x  
    print(a, b, x, y)  
    return x  
  
a, b, x, y = 1, 2, 3, 4  
x = foo(17, 4)  
print(a, b, x, y)
```

Answer + Explanation:

A:

```
42 2 4 17  
1 2 3 4
```

a, b, x and y are all global variables. In the function `foo()`, `a` is overshadowed by local variable `a` and `x` & `y` are overshadowed by the parameters `x` and `y`. `b` references the global variable as there is no variable `b` declared inside the function.

No change is made to global variables `a, b, x` or `y` since the only changes `foo()` can make are to its internal local variables (while they're all immutable types).

Question 3, what is the output of this code?

3. What is the output of this code? Why?

```
def mystery(x):  
    x.append(5)  
    x[0] += 1  
    print("mid-mystery:", x)  
  
my_list = [1, 2]  
print(my_list)  
mystery(my_list)  
print(my_list)  
mystery(my_list.copy())  
print(my_list)
```

A:

```
[1, 2]  
mid-mystery: [2, 2, 5]  
[2, 2, 5]  
mid-mystery: [3, 2, 5, 5]  
[2, 2, 5]
```

Answer +
Explanation:

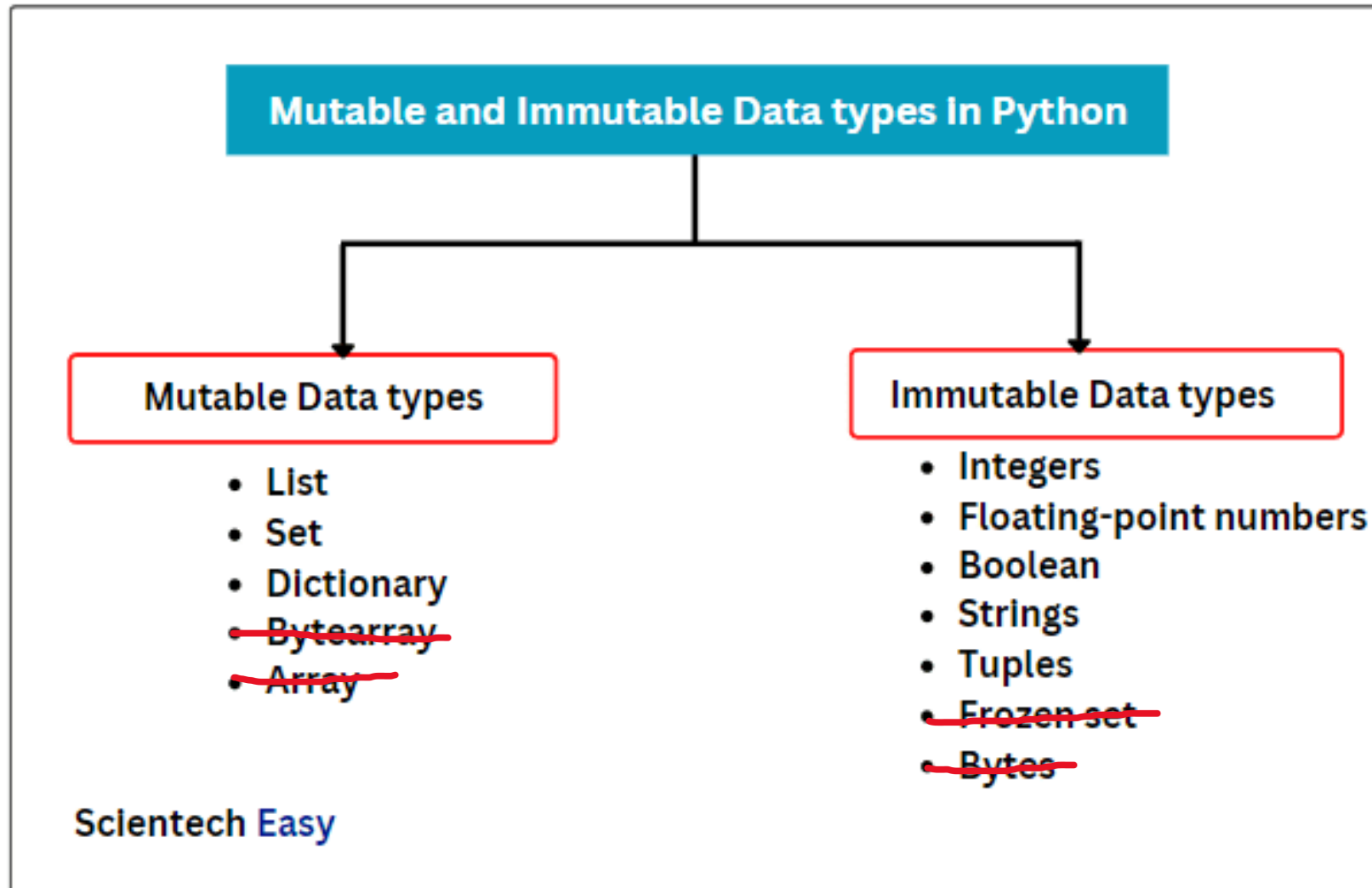
The `mystery()` function is a mutating function, as it changes the elements of the list argument it is given.

In the first example, the function is called with the original list and therefore it is mutated: the value printed mid-mystery is reflected outside the function.

The second time, a copy of the original list is passed as an argument, meaning any change made inside the function are not applied to the list, as we can see with how the last two lines printed are different.

If the `mystery()` function were changed to alter a string, we would see that no matter how a string is passed into a function, it can never be mutated because it is an immutable type. Returning a new string is the only way to pass changes outside a function.

Revision: Mutable and Immutable Data Types



Question 4, compare and contrast the 2 functions

4. Compare the two functions below. Are they equivalent? Why would we prefer one over the other?

```
def noletter_1(words, letter='z'):
    for word in words:
        if letter in word:
            return False
    return True

def noletter_2(words, letter='z'):
    no_z = True
    for word in words:
        if letter in word:
            no_z = False
    return no_z

wordlist = ['zizzer'] + ['aardvark'] * 10_000_000
print(noletter_1(wordlist))
print(noletter_2(wordlist))
```

A: The two functions are functionally equivalent, but the first one uses a timely return while the second one doesn't. In the example of 'zizzer' followed by 10 million instances of 'aardvark', the first function will perform much faster as it's able to return False as soon as it tests 'zizzer'. The second will continue to iterate through every instance of 'aardvark' before returning False, taking much more time unnecessarily.

Explanation:

Programming on Paper



Past MST Question 2023 Semester 1

1. Write a Python function `find_ints(text)` that takes a (possibly empty) Python string `text` and returns a (possibly empty) list of the word locations at which integers occur. A “word” is defined as a consecutive sequence of non-whitespace characters, and an “integer” is defined as a word that is either completely made up of digits, or is a single + or – sign, and then nothing but digits. Word positions within text are counted from one.

For example:

- `find_ints("Ints -34 there and here +551 but not here88")` should return `[2, 6]`
- `find_ints("No integers here99 88-77 or there")` should return `[]`
- `find_ints("+18 and -777 and 666 are all 3 integers")` should return `[1, 3, 5, 8]`

Hint: if you get stuck, try separating the first character from the rest for each word.

A:

```
def find_ints(text):  
    int_pos = []  
    words = text.split()  
    for i in range(len(words)):  
        word = words[i]  
        if word.isdigit() or word[0] in "+-" and word[1:].isdigit():  
            int_pos.append(i+1)  
    return int_pos
```

Answer:

Exercise 1 / 3

1. Write a function which takes two lists as input and returns a list containing the numbers which they both have in common. `in_common([1, 2, 4], [3, 4, 5])` should return `[4]`.

Answer:

A:



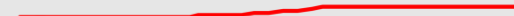

```
def in_common(list_1, list_2):  
    set_1 = set(list_1)  
    set_2 = set(list_2)  
    common = set_1 & set_2  
    return list(common)
```

Exercise 2 / 3

2. Write a function which takes a dictionary and returns a sorted list containing the unique values in that dictionary. `unique_values({'a': 1, 'b': 0, 'c': 0})` should return `[0, 1]`.

Answer:

A: Note that this solution only works if the values of the input dictionaries are immutable, since the values in a set must be immutable.

```
def unique_values(d):  
    values = d.values()   
    set_values = set(values)   
    sorted_values = sorted(set_values)   
    return sorted_values 
```

Exercise 3 / 3

3. Write a function which takes a list of words and checks whether any of those words are palindromes (spelled the same way backwards as forwards, like “kayak”). It should return True if there are any palindromes and False if there are none. Use a timely return to save some time!

`any_palindrome(['kayak', 'motorbike', 'scooter'])` should return True.

Answer:

A:

```
def any_palindrome(words):  
    for word in words:  
        → if word == word[::-1]:  
            return True  
    return False
```

Independent Work

- **MST page** on canvas has been released!
 - It contains **where and when** exactly is your test.
 - It also contains **9 past papers**, 3 of which has sample solutions.
 - **Practice, practice, practice!!!**
- **NO Ed Worksheets due next week.**
 - Please focus on your **Project 1**. This is **due Friday, May 2nd, 6pm.**
- **Raise your hand** if you have any questions!

Scan here for annotated slides

