Week 6 Tutorial

COMP10001 – Foundations of Computing

Semester 2, 2025

Clement Chau

- sorted(list) vs. list.sort()
- Global and Local Namespaces (Scope)
- Early returns

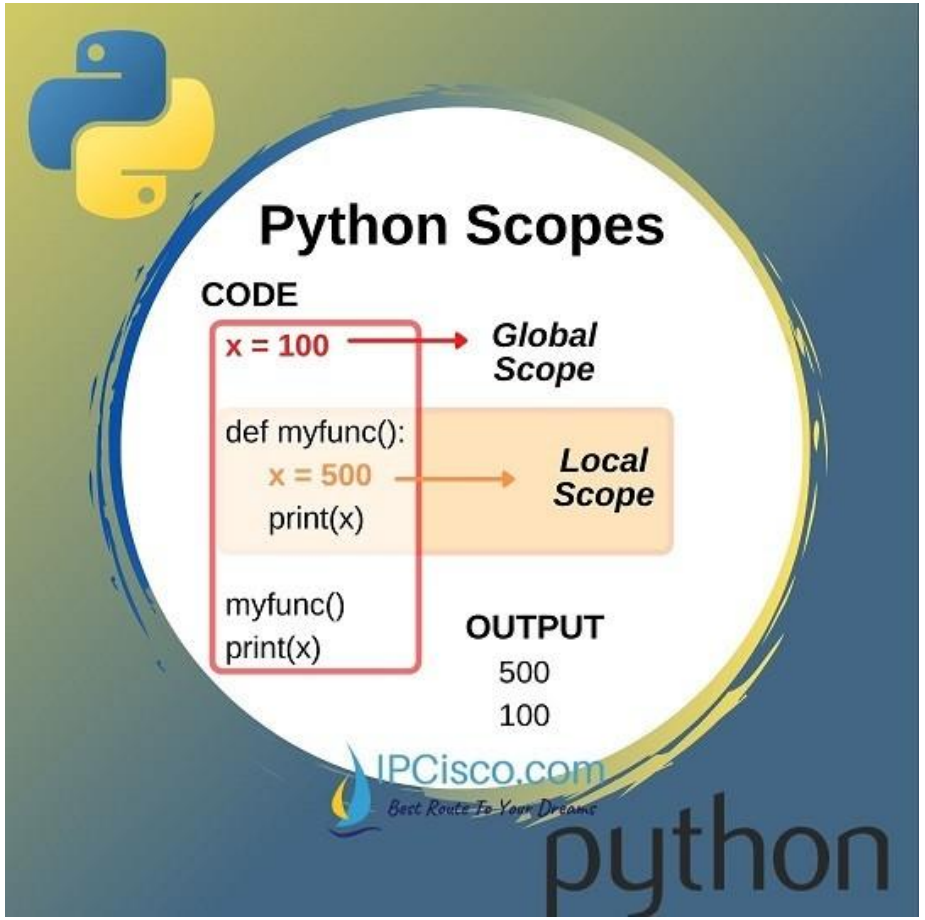
Agenda

1. Week 6 Discussion – **Tutorial sheet** (~ 55 mins)
2. One-on-one Q&A for **Ed lessons, Project 1** (~ 55 mins)

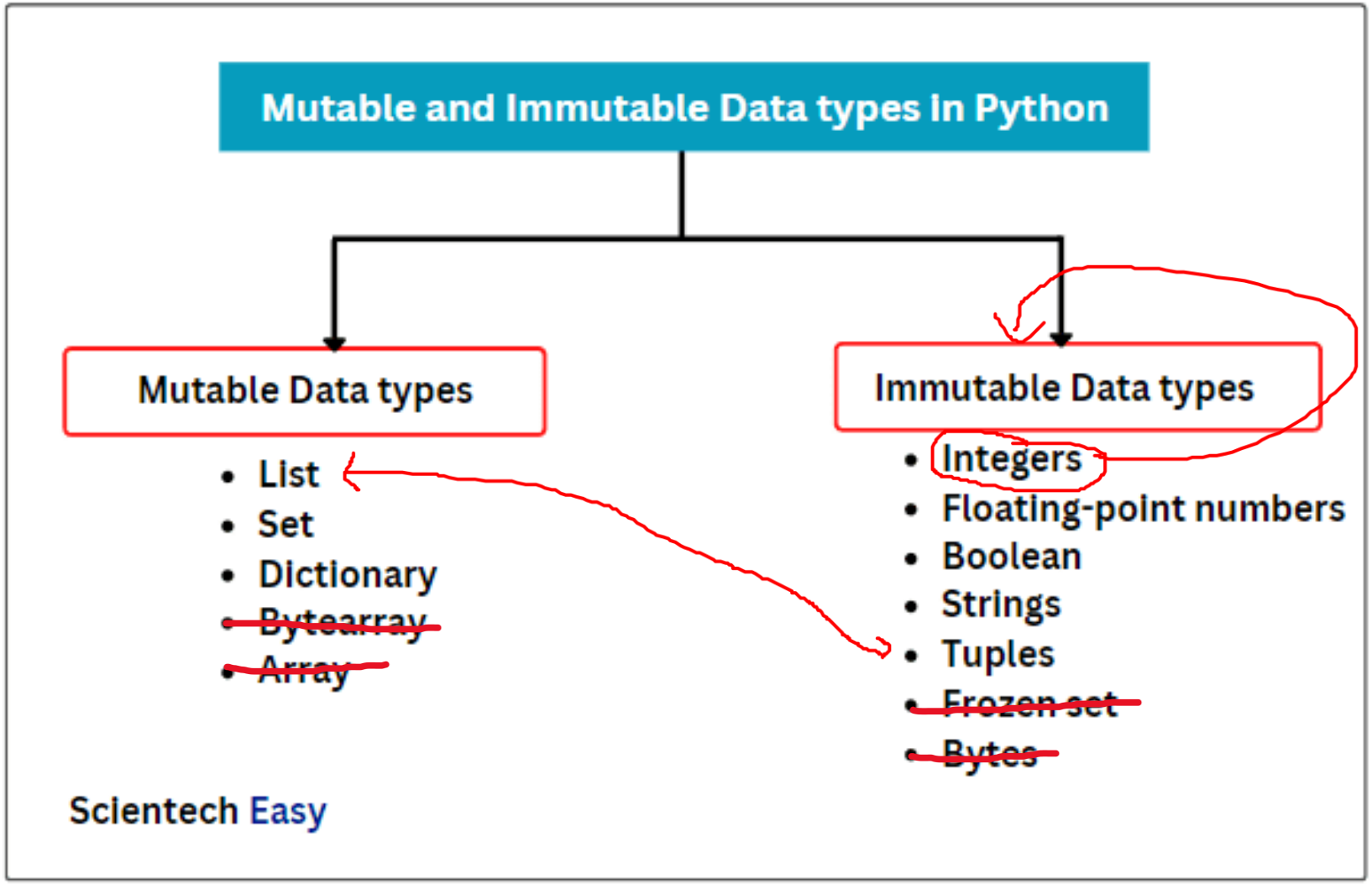
6 (1/9)	Code readability and debugging, errors	Debugging and Testing; Libraries	Mid-Semester Test preparation	Week 6 tutorial sheet Week 6 tutorial solutions	• Ed worksheet 9 due (1/9 at 6 pm)

Ed worksheets 9 due (1/Sep, Monday at 6 pm)
Mid-Semester Test (11/Sep, Thursday at Lecture time)

Revision: Python Scopes!



Revision: Mutable and Immutable Data Types



Useful Tips for Your Assignments

- **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.

TuteSheet Week 6 – Question 1 (a)

1. Assume that `seq` is a list. Write one line of code to:

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

`seq.sort()` *Python lists have a built-in* `list.sort()` ***method*** *that* ***modifies*** *the list* ***in-place.*** *Returns* ***None.***

What if **`seq = seq.sort()`** ***?***

```
>>> seq = [4, 1, 3, 2, 5]
>>> seq.sort()
>>> print(seq)
[1, 2, 3, 4, 5]

>>> seq = seq.sort()
>>> print(seq)
None
```

returns None

https://docs.python.org/3/howto/sorting.html

TuteSheet Week 6 – Question 1 (b)

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

```
new_seq = sorted(seq)
```

*There is also a `sorted()` **built-in function** that builds a new sorted list from an iterable.*

What if `new_seq = sort(seq)`

```
>>> new_seq = sorted(seq)
>>> print(new_seq)
[1, 2, 3, 4, 5]

>>> new_seq = sort(seq)
NameError
```

TuteSheet Week 6 – Question 1 (c)

(c) Add the string "hi" to the end of seq.

```
seq = seq + ["hi"]
seq.append("hi")
```

What if `seq = seq.append("hi")`

```
>>> seq = [4, 1, 3, 2, 5]
>>> seq.append("hi") # modifies seq in place
>>> print(seq)
[4, 1, 3, 2, 5, 'hi']

>>> seq = seq.append("hi") # seq now becomes None
>>> print(seq)
None
```

TuteSheet Week 6 – Question 2

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)        42 2 4 17

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

1) Output

2) Variables

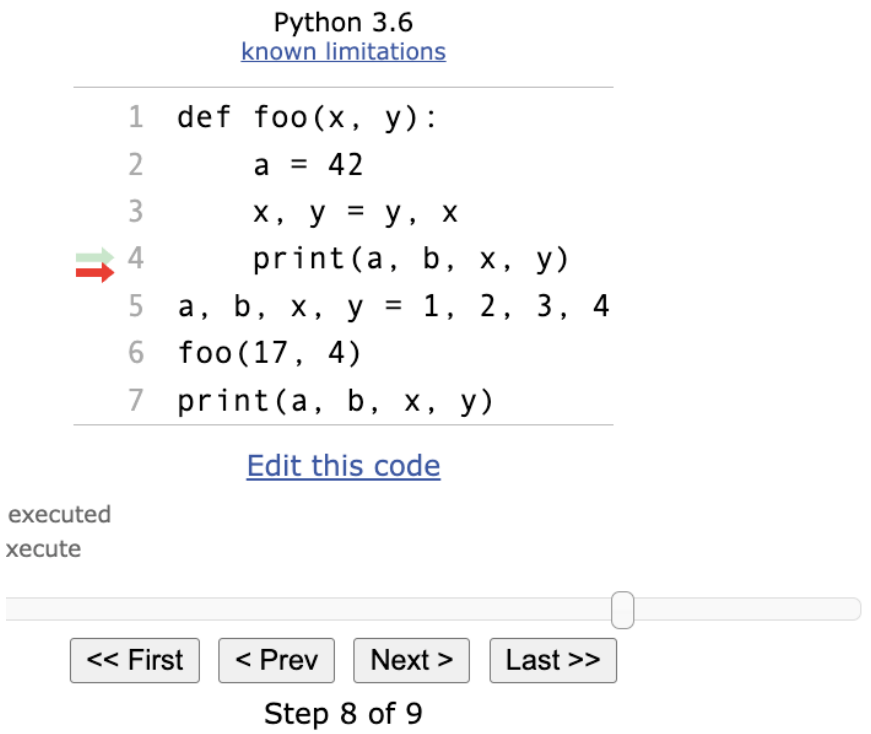
- ***Global*** `a, b, x, y`

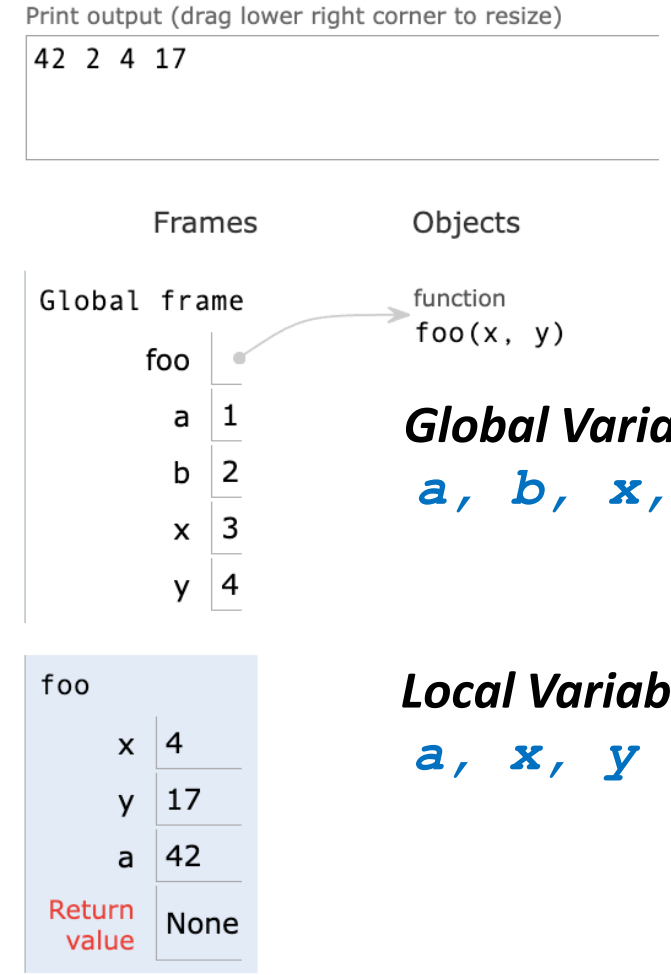
*No change is made to global variables **a, b, x** or **y** since the only changes foo() can make are to its internal local variables*

- ***Local*** `a, x, y`

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

TuteSheet Week 6 – Question 2





Global Variables

a, b, x, y

Local Variables

a, x, y

https://pythontutor.com/

TuteSheet Week 6 – Question 3

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)
```

Pair Activity

For Pairs

- *One person: **Examiner** (try running the code in Python Tutor)*
- *Other person: **Examinee** (solves question and answer using pen/paper)*

In 3 mins,

- *Examiner tries to **understand** the solution.*
- *Examinee reads and **solves** the question.*

In 2 mins,

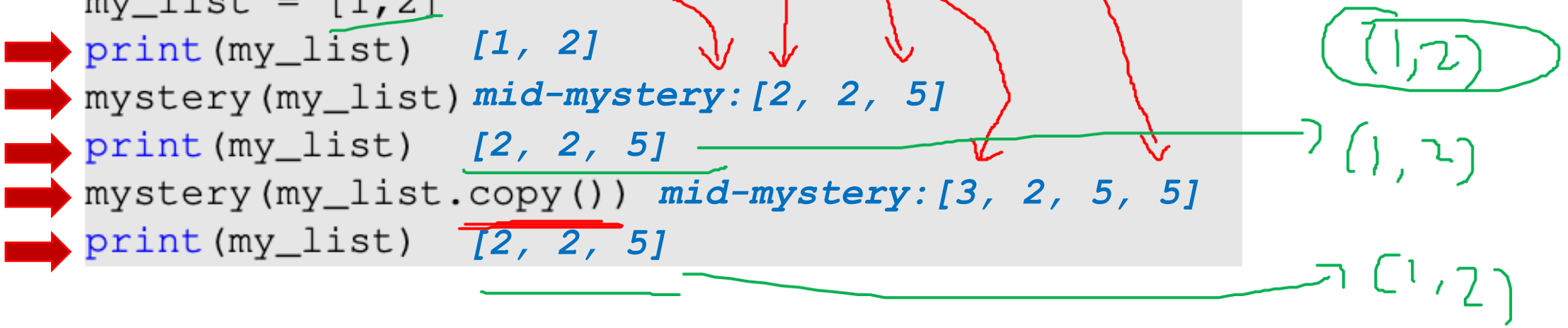
- *Examiner listens and **assesses** the explanation.*
- *Examinee **explains** the answer.*

TuteSheet Week 6 – Question 3

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)                  [1, 2]
mystery(my_list)                mid-mystery:[2, 2, 5]
print(my_list)                  [2, 2, 5]
mystery(my_list.copy())         mid-mystery:[3, 2, 5, 5]
print(my_list)                  [2, 2, 5]
```



TuteSheet Week 6 – Question 3

Python 3.6
known limitations

```
1  def mystery(x):
2      x.append(5)
3      x[0] += 1
4      print("mid-mystery:", x)
5
6  my_list = [1,2]
7  print(my_list)
8
9  mystery(my_list)
10 print(my_list)
11
12 mystery(my_list.copy())
13 print(my_list)
```

Edit this code

uted
te

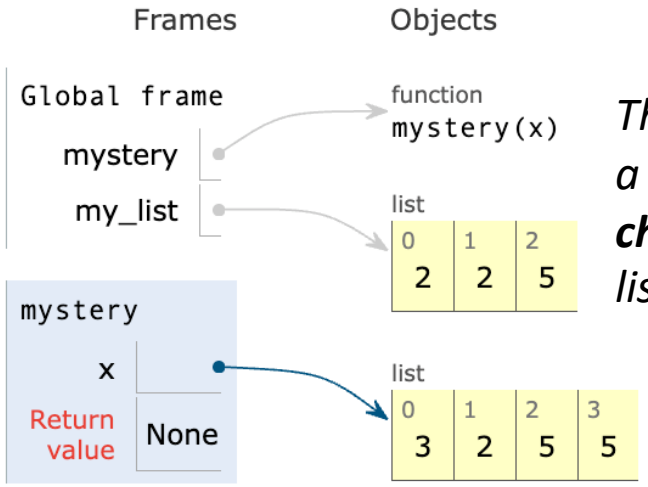
<< First | < Prev | Next > | Last >>

Step 16 of 17

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

Print output (drag lower right corner to resize)

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



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

*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 original list, as we can see with how the last two lines printed are different.*

TuteSheet Week 6 – Question 4

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))
```

*The two functions are **functionally equivalent**, but the first one uses a timely return while the second one doesn't*

*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.*

['zizzer', 'aardvark', 'aardvark', …, 'aardvark']

TuteSheet Week 6 – Exam Practice

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]`

     `['Ints', '-34', 'there', 'and', 'here', '+551', 'but', 'not', 'here88']`

   - `find_ints("No integers here99 88-77 or there")` should return `[]`

     `['No', 'integers', 'here99', '88-77', 'or', 'there']`

   - `find_ints("+18 and -777 and 666 are all 3 integers")` should return `[1, 3, 5, 8]`

     `['+18', 'and', '-777', 'and', '666', 'are', 'all', '3', 'integers']`

TuteSheet Week 6 – Exam Practice

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**.

```
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
```

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**. It's **due Friday, September 19th, 6pm**.
- **Raise your hand** if you have any questions!

Scan here for annotated slides