



CS1117 – Introduction to Programming

Dr. Jason Quinlan,
School of Computer Science and Information Technology

A TRADITION OF
INDEPENDENT
THINKING



University College Cork, Ireland
Coláiste na hOllscoile Corcaigh



► LiveSlides web content

To view

Download the add-in.

liveslides.com/download

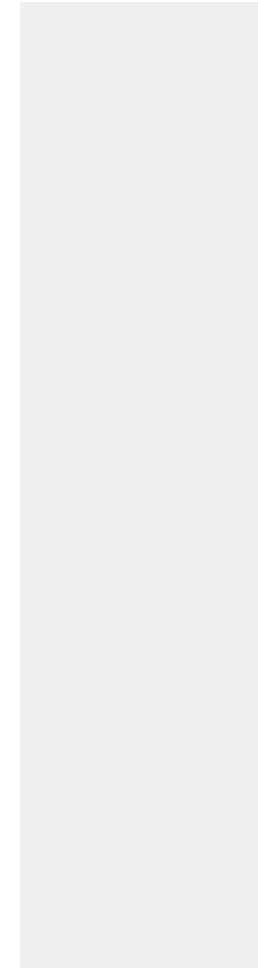
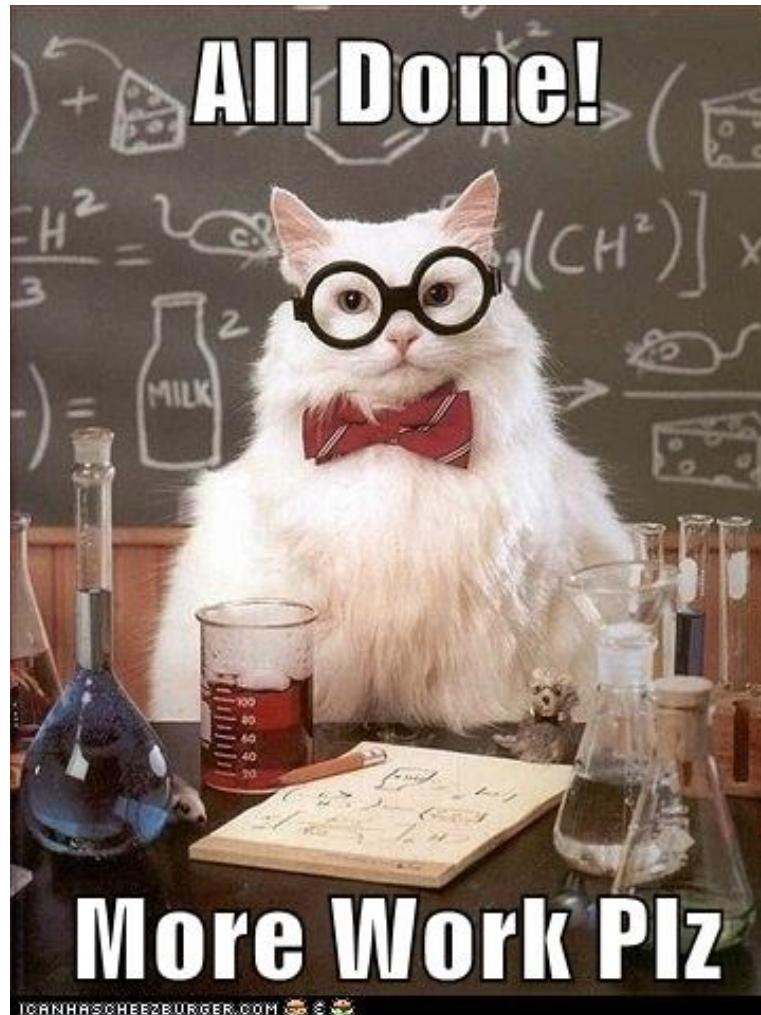
Start the presentation.

Canvas Student App



Let's Sign into this lecture now

Access Code
68346



Dictionary Recap

Is there any questions on Dictionaries from Monday?

Dictionary Example

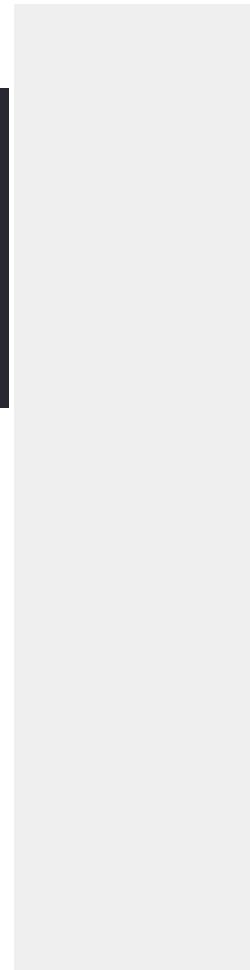


Some more Dictionary examples:

```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
```



```
# month of birth
mob = {'January': ['aishling', 'tim'], 'February': ['bob']}
```



Dictionary Example

Some more Dictionary examples:

```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
```



```
# month of birth
mob = {'January': ['aishling', 'tim'], 'February': ['bob']}
```

```
# print month:
for m in mob:
    print(m)
# prints
# January
# February

for m in mob.keys():
    print(m)
# prints
# January
# February
```

Dictionary Example



Some more Dictionary examples:

```
# date of birth  
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
```



```
# month of birth  
mob = {'January': ['aishling', 'tim'], 'February': ['bob']}
```

```
# print month:  
for m in mob:  
    print(m)  
# prints  
# January  
# February  
  
for m in mob.keys():  
    print(m)  
# prints  
# January  
# February
```

```
# print people in month:  
for m in mob:  
    print(mob[m])  
# prints  
# ['aishling', 'tim']  
# ['bob']  
  
# print people in month:  
for v in mob.values():  
    print(v)  
# prints  
# ['aishling', 'tim']  
# ['bob']
```

Dictionary Example

Some more Dictionary examples:

```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}

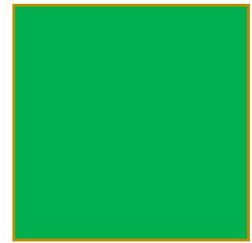
# month of birth
mob = {'January': ['aishling', 'tim'], 'February': ['bob']}

months = {1: 'January', 2: 'February'}
```



```
def to_month(dob):
    ...
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months as keys and
    a list of names born in those months as values
    ...
    return
```

Dictionary Example



Some more Dictionary examples:

```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
```



```
# month of birth
# mob = {'January': ['aishling', 'tim'], 'February': ['bob']}
```



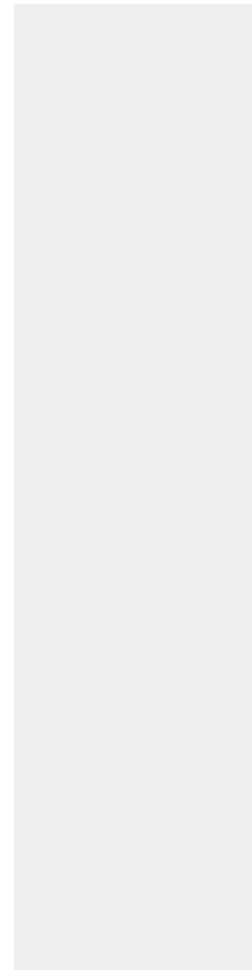
```
def to_month(dob):
    ...
    dob – take in a dictionary of peoples date of birth
    return – a dictionary of months keys and
    a list of names born in those months as values
    ...
months = {1: 'January', 2: 'February'}
```



```
return
```



```
# month of birth
mob = to_month(dob)
```



Dictionary Example

Some more Dictionary examples:

```
def to_month(dob):
    """
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    """

    months = {1: 'January', 2: 'February'}
```

```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# None
```

Dictionary Example

Some more Dictionary examples:

```
def to_month(dob):
    ...
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    ...
months = {1: 'January', 2: 'February'}
```



```
# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
```

Dictionary Example

Some more Dictionary examples:

```
def to_month(dob):
    """
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    """

    months = {1: 'January', 2: 'February'}
    mob_dict = {}

    return mob_dict

# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# {}
```

Dictionary Example

Some more Dictionary examples:

```
def to_month(dob):
    """
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    """
    months = {1: 'January', 2: 'February'}
    mob_dict = {}
    for key, val in dob.items():
        print(key, val)

    return mob_dict

# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# aishling 1/1/1995
# bob 2/2/1996
# tim 17/1/1995
# {}
```

Dictionary Example

```
def to_month(dob):
    """
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    """
    mob_dict = {}
    for key, val in dob.items():
        month = get_month(val)
        print(month)

    return mob_dict

# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# January
# February
# January
# {}
```

Dictionary Example

```
def get_month(val):
    """
    take a dob in european format - '1/1/1995'
    return integer for month - 1 -> 'January', etc.
    """

    months = {1: 'January', 2: 'February'}

    slash_first_index = val.index("/") # 1
    val = val[slash_first_index+1:] # '1/1995'

    slash_second_index = val.index("/") # 1
    val = val[:slash_second_index] # '1'

    val = int(val) # 1

    return months[val] # 'January'
```

Dictionary Example

```
def to_month(dob):
    """
    dob - take in a dictionary of peoples date of birth
    return - a dictionary of months keys and
    a list of names born in those months as values
    """

    mob_dict = {}
    for key, val in dob.items():
        month = get_month(val)
        mob_dict[month] = [key]

    return mob_dict

# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# {'January': ['tim'], 'February': ['bob']}
```

Dictionary Example

```
def to_month(dob):
    ...
    dob – take in a dictionary of peoples date of birth
    return – a dictionary of months keys and
    a list of names born in those months as values
    ...
    mob_dict = {}
    for key, val in dob.items():
        month = get_month(val)
        if month in mob_dict:
            mob_dict[month].append(key)
        else:
            mob_dict[month] = [key]

    return mob_dict

# date of birth
dob = {'aishling': '1/1/1995', 'bob': '2/2/1996', 'tim': '17/1/1995'}
# month of birth
mob = to_month(dob)
print(mob)
# {'January': ['aishling', 'tim'], 'February': ['bob']}
```

Lab Recap

Let us look at some of the issues that keep popping up in the lab submissions:

Let's start with **break** and **return**:

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

The instruction was to create a function called `index` which takes 2 parameters (`list, value`) and return the first index of `value` or `-1` if index does not exist in the list

so this code is fine ☺

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

But, I've seen people write `list.index(value)` in their code 😞

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

Let's look at the return -1

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

Let's look at the return -1

This will never be returned – why??

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

Let's look at the return -1

This will never be returned – why??

1. If value in list – returns True – then return -1 is never called

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

Let's look at the return -1

This will never be returned – why??

2. The else: is indented too far – it should line up to if

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

Let's look at the return -1

This will never be returned – why??

3. Not an issue here, but a call to break will not call else after while...

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
    else:
        return -1
```

So let's move the return -1

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
|
    return -1
```

So let's move the return -1

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
|
    return -1
```

So let's move the return -1

Remember if value in list – return -1 never called

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            break
            i += 1
|
    return -1
```



So let's move the return -1

Remember if value in list – return -1 never called

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    if value in list:
        i = 0
        while i < len(list):
            if list[i] == value:
                return i
            i += 1
    return -1
```

So let's move the return -1

Remember if value **not** in list – return -1 is called

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
        break
        i += 1

    return -1
```

Actually we don't need the `if value in list` (but it does speed up our code...)

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
            break
        i += 1

    return -1
```

Finally, let's look at the `return` and `break`

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
        break
        i += 1

    return -1
```

return ends the function (may return a value to the calling code)

Break stops and exits the loop (ignoring else)

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
        break
        i += 1

    return -1
```

return ends the function (may return a value to the calling code)

Break stops and exits the loop (ignoring else)

So what's the problem here?

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
        break
        i += 1

    return -1
```

return ends the function (may return a value to the calling code)

Break stops and exits the loop (ignoring else)

So what's the problem here? – break is never called

Lab Recap

This is the index function from Lab 6

```
def index(list, value):
    # if value in list:
    i = 0
    while i < len(list):
        if list[i] == value:
            return i
            # break
        i += 1

    return -1
```

return ends the function (may return a value to the calling code)

Break stops and exits the loop (ignoring else)

So what's the problem here? – so remove break

Lab Recap

Let us look at some of the issues that keep popping up in the lab submissions:

Let's now look at **shadowing**:

Shadowing is redefining a variable/function name which masks the original use of the name

Lab Recap

Index() from Lab 6

```
def index(list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

Lab Recap

```
def index(list, value):
    # set counter to 0
    i = 0
    # loop over the list
    while i < len(list):
        # if current element is the same as the value
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

Built-in mutable sequence.

If no argument is given, the constructor creates a new empty list.

The argument must be an iterable if specified.

Lab Recap

```
def index(list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

Lab Recap

```
def index(list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
    # increment the counter
    i += 1
return index(value)
```

Return first index of value.

Raises ValueError if the value is not present.

Lab Recap

```
def index(list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

```
index(list, value)      ↗
def index(list, value): ↗
list = [1, 2, 3]         ↗
```

Lab Recap

If I change the `list` parameter to `my_list`

```
def index(my_list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

No errors!!!

Lab Recap

If I change the `value` parameter to `my_value`

```
def index(my_list, my_value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(list):
        # if <value> and <list> index i are the same
        if list[i] == value:
            return list.index(value)
        else:
            print('-1')
        # increment the counter
        i += 1
    return index(value)
```

Errors!!!

Lab Recap

Rewrite the function

```
def index(my_list, value):
    # set counter
    i = 0
    # loop over the length of the <list>
    while i < len(my_list):
        # if <value> and <list> index i are the same
        if my_list[i] == value:
            return i
        # increment the counter
        i += 1
    # value not in list, so return -1
    return -1
```

Return i, not list.index(value)

Move and return -1, not “-1”

Lab Recap

1 more example:

```
def index(list, value):
    # create an empty list
    index_val = []
    # starting index
    idx = -1
    # while True – continue for ever
    while True:
        # call list index
        idx = list.index(value, idx + 1)
        # add the found index to the list
        index_val.append(idx)
        # break the while loop
        break
    # return the index
    return index_val
```

Lab Recap

1 more example:

```
def index(list, value):
    # create an empty list
    index_val = []
    # starting index
    idx = -1
    # while True – continue for ever
    while True:
        # call list index
        idx = list.index(value, idx + 1)
        # add the found index to the list
        index_val.append(idx)
        # break the while loop
        break
    # return the index
    return index_val
```

```
my_list = [1, 2, 12]
print(index(my_list, 12))
# [2]
print(index(my_list, 13))
# ValueError: 13 is not in list|
```

Lab Recap

2 examples of coding till it works 😊

```
def value_in_list(list1, value):
    while list1 != value:
        if value in list1:
            return True
    else:
        return False
```

```
def get_value(list, index):
    while index < len(list):
        return list[index]
    return list[index]
```

Lab Recap

Let us look at some of the issues that keep popping up in the lab submissions:

A few other things to keep in mind:

Lab Recap

Print is not return

Print is only used so we can see what a variable value is

Or what a value returned from a functions is

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

The Python `None` keyword is used to define a null value, or no value at all

`None` is not the same as `0`, `False`, or an empty string

`None` is a datatype of its own (`NoneType`) and only `None` can be `None`

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
# prints 'None'
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1

    return var

print(what_is_returned())
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1

    return var

    print(what_is_returned())
# prints 1
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1

    return var

print(what_is_returned())
# prints 1
```

```
def what_is_returned():
    var = 1

    return var

what_is_returned()
```

Lab Recap

```
def what_is_returned():
    var = 1
    -----
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1
    -----
    return
    |
    print(what_is_returned())
# prints 'None'
```

```
def what_is_returned():
    var = 1

    return var

print(what_is_returned())
# prints 1
```

```
def what_is_returned():
    var = 1

    return var

what_is_returned()
# nothing is presented on terminal
# or in the run screen
```

Lab Recap

Let us look at some of the issues that keep popping up in the lab submissions:

Let's now look at lists and slicing:

Lab Recap

`start = [:index]`

is not

`start = list[:index]`

Also, remember

`list[:index]` stops at the value of `index-1`

Lab Recap

```
my_list = [4, 5, 6, 7]
max_index = 3
sub_list = [:max_index]
sub_list = my_list[:max_index]
```

Lab Recap

```
my_list = [4, 5, 6, 7]
max_index = 3
sub_list = [:max_index]
sub_list = my_list[:max_index]
```

```
my_string = "hello"
max_index = 3
sub_string = [:max_index]
sub_string = my_string[:max_index]
```

Lab Recap

```
my_list = [4, 5, 6, 7]
max_index = 3
sub_list = [:max_index]
sub_list = my_list[:max_index]
```

```
my_string = "hello"
max_index = 3
sub_string = [:max_index]
sub_string = my_string[:max_index]
```

```
my_string = "hello"
max_index = 3
# sub_string = [:max_index]
sub_string = my_string[:max_index]
print(sub_string)
# hel
```

Lab Recap

`list1.append(list2)`

Will add list2 as a single element to list1:

`[1, 2, 3].append([2, 3, 4])`

`[1, 2, 3, [2, 3, 4]]`

