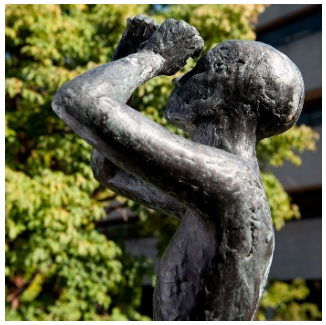


CS1117 – Introduction to Programming

Dr. Jason Quinlan,

School of Computer Science and Information Technology

A TRADITION OF INDEPENDENT THINKING

UCC

University College Cork, Ireland

Coláiste na hOllscoile Corcaigh

Announcements

CS1117 Labs start this week:

Labs		
Tues (17th Sept.)	4 - 6pm	G.20
Wed (18th Sept.)	4 - 6pm	G.20

Student Allocation for the Labs have been posted to Canvas

Announcements

Go to Canvas

You will find the lab allocation under module "Week 2”

View "lab allocation CS1117.pdf”

Make a note of which day you are allocated too.

Announcements

Very Important

All BSc DSA students must attend the Tuesday 4 - 6pm lab

All EC1202 students must attend the Tuesday 4 - 6pm lab

All FR0105 students must attend the Wednesday 4 - 6pm lab

Other than those students, IF you have a valid reason not to attend the allocated session, contact CS Admin

Announcements

Note

Approximately 140 in this class

Max number in G20 is 80 students
(assuming all machines are working)

At most only 10 can move…

Announcements

Note

You will need your CS account details to log in to the PCs

So, go to G20 and make sure you log in

IF you can't go to 1.25 CS IT admin and they will help

Announcements

Note

Initially, the Labs will follow the content of the lectures

A few questions on the content of each lecture

So I know you understand the lectures…

Announcements

Note

Lab assignments will be pushed to Canvas tomorrow

Submission deadline is this coming Saturday @ 1am

I goal is to have the submissions graded by next Lab

I'll also add comments to the assignment on Canvas as we progress through the semester

Announcements

Volunteer Research Assistant

How do I apply for this amazing opportunity?

Email me: j.quinlan@cs.ucc.ie

Put “MISL VRA S1 19” in the subject line

Deadline of today, 16th Sept, @ 12 midnight

Quick 5 minute meetings to be arranged from tomorrow

Final decision by others ☺

Canvas Student App

Download the Canvas Student App

Canvas Student

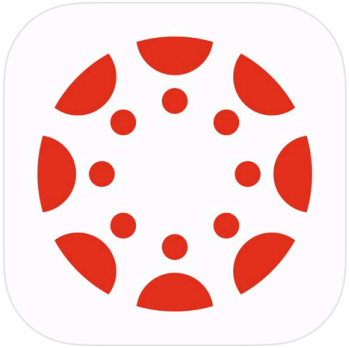
Instructure Education

PEGI 3

You don't have any devices.

Add to wishlist

Android



Canvas Student 4+

Instructure Inc.

#11 in Education

★★★★★ 4.7, 924.5K Ratings

Free

iOS

Take a moment now, to download and setup Canvas on your device

University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Canvas Student App

- **Download** the Canvas Teacher/Student app from the Play Store/App Store on Android/iOS devices.
- **Open the app**. You will be presented with a page to search for your school.
- Tap **Find Your School.**
- **Enter "University College Cork"** and hit the arrow to continue.
- This will prompt you to enter your username and password
- You must **authorise** that you are allowing Canvas to access your account.
- **Log in with your regular credentials** and you'll see your list of courses.

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

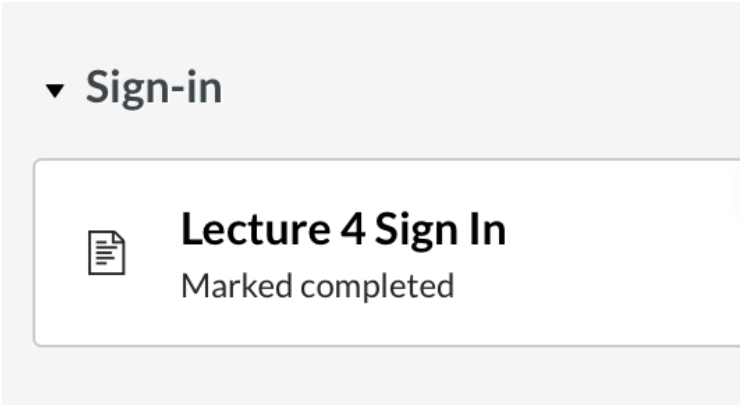
Canvas Student App

I added a new module called “Sign-in”

Canvas Student App

I added a new module called “Sign-in”

At some stage during the class, I’ll ask you to click on todays lecture (Lecture 4 Sign In)

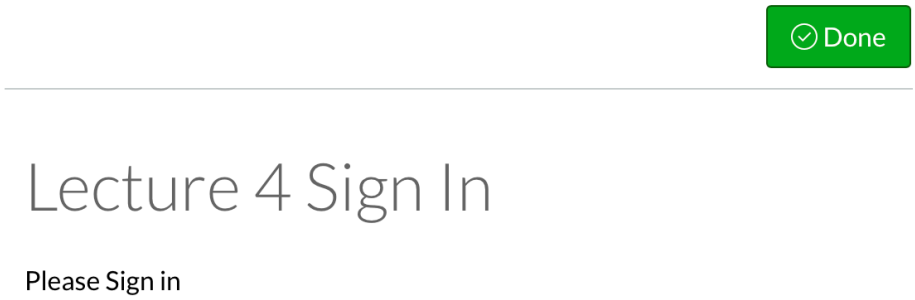


UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Canvas Student App

I added a new module called “Sign-in”

At some stage during the class, I’ll ask you to login click on todays lecture (Lecture 4 Sign In) and click Done



Canvas Student App

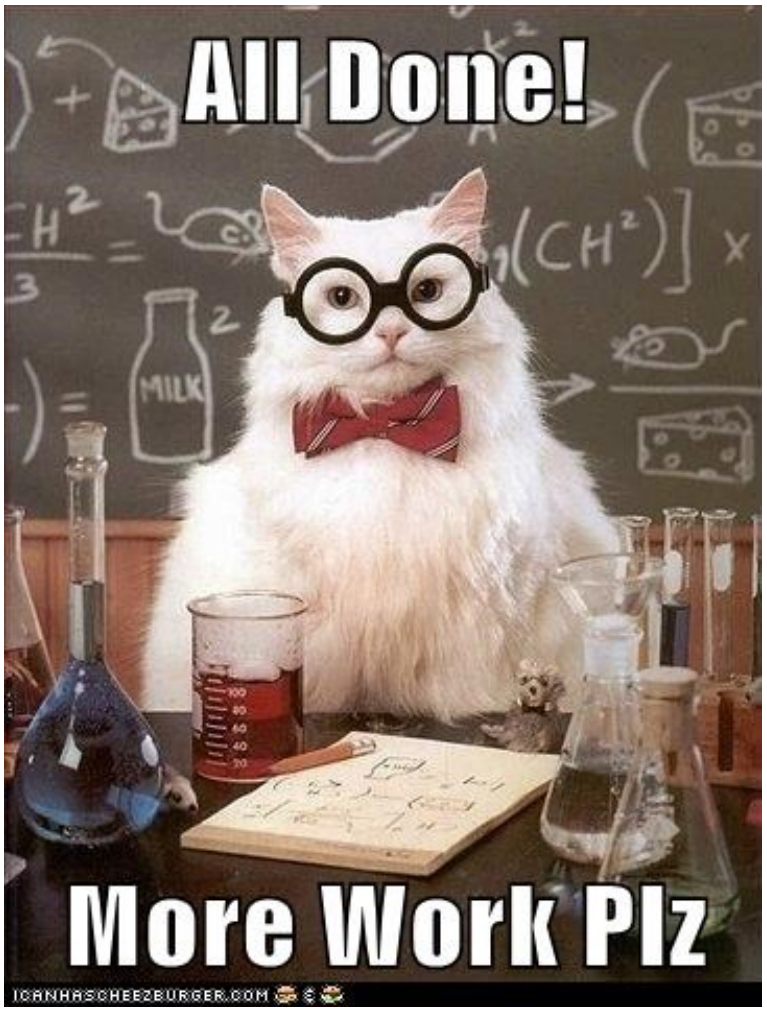
I added a new module called “Sign-in”

If this works, no more paper sign-in’s ☺

And I’ll have the sign in for the class at a given time during the lecture

Canvas Student App

Let’s Sign into this lecture now



UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh

Python Variables Recap

So back to Python, previously…

- We assign a variable a value
- Data types (integer, float, boolean, tuple, list, dictionary)
- Variable naming
  - Reserved words
  - Ambiguity in naming - I (i) and l similarity
  - Camel and Pascal case naming
  - Do not use a number as first character (7_weeks)
  - Do not use "#" in the variable name

Python Variables/Functions

- As stated two main building blocks for Code:

- Variables, which we just covered
  - It's a means of assigning (giving) a bit of information to a name, that we can use later in the code
  - Saves on repeating the original code used to create the value.
  - We just use the name and get the original value
  - age = 7
  - print(age) -> prints 7 to the screen

- Functions:
  - Similar to variables, as it's a means of reducing the amount of code you write
  - But, typically not for single lines of code, but for a number of lines of code (statement block)

Python Functions

- Example:

- Let's go back to our "average_age" code

```
# averge ages:
# get the first age
age1 = int(input("Please enter age 1: "))
# get the second age
age2 = int(input("Please enter age 2: "))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions

- Let's rewrite input() to take a variable rather than a string:

```
# averge ages - remove the strings from input()
# and create seperate variables for the strings
# get the first age
input_string_1 = "Please enter age 1: "
age1 = int(input(input_string_1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = int(input(input_string_2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions

- Let's rewrite input() to take a variable:

```
# averge ages - remove the strings from input()
# and create seperate variables for the strings
# get the first age
input_string_1 = "Please enter age 1: "
age1 = int(input(input_string_1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = int(input(input_string_2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

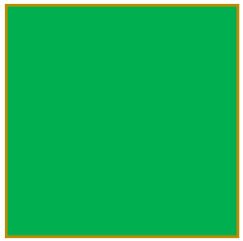
Python Functions

- Let's rewrite input() to take a variable:

```
# averge ages - remove the strings from input()
# and create seperate variables for the strings
# get the first age
input_string_1 = "Please enter age 1: "
age1 = int(input(input_string_1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = int(input(input_string_2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

- Code repetition – very similar code with the same task/job.

Python Functions



- Let's define a function called "ask_for_int_input"

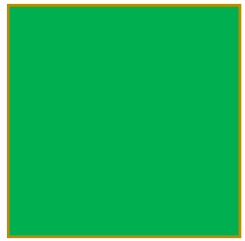
Python Functions

- Let's define a function called "ask_for_int_input"

def – keyword for Python function

def

Python reserved keywords



Just known that within Python

these 33 reserved keywords exist

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

Python Functions

- Lets define a function called “ask_for_int_input”

def – keyword for Python function

def

Python Functions

- Lets define a function called “ask_for_int_input”

def – keyword for Python function

Function name ask_for_int_input

```
def ask_for_int_input()
```

Function has opening and closing parentheses **(round brackets)**

Python Functions

- Lets define a function called “ask_for_int_input”

def – keyword for Python function

Function name ask_for_int_input

Parameter(s) passed to the function question_string

```
def ask_for_int_input(question_string)
```

Python Functions

- Lets define a function called “ask_for_int_input”

def – keyword for Python function

Function name ask_for_int_input

Parameter(s) passed to the function question_string

```
def ask_for_int_input(question_string)
```

A function can have zero or more parameters

Python Functions

- Lets define a function called “ask_for_int_input”

def – keyword for Python function

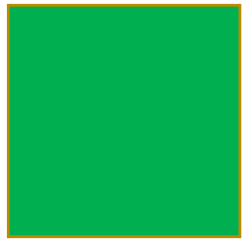
Function name ask_for_int_input

Parameter(s) passed to the function question_string

```
def ask_for_int_input(question_string):
```

Function definition ends in “colon” :

Python Functions



- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

Python Functions

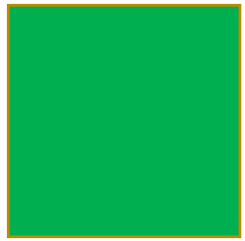
- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

- And we know what we want the function to do:

```
user_input = int(input(question_string))
```

Python Functions



- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

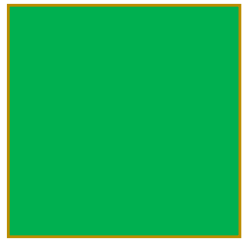
- And we know what we want the function to do:

```
user_input = int(input(question_string))
```

- So, let's put these together:

```
def ask_for_int_input(question_string):
    user_input = int(input(question_string))
```

Python Functions



- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

- And we know what we want the function to do:

```
user_input = int(input(question_string))
```

- So, let's put these together:

```
def ask_for_int_input(question_string):
    user_input = int(input(question_string))
```

- Notice the indentation?

Python Functions

- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

- And we know what we want the function to do:

```
user_input = int(input(question_string))
```

- So, let's put these together:

```
def ask_for_int_input(question_string):
    user_input = int(input(question_string))
```

4 spaces

- Notice the indentation?

Python Functions

- Now we have our function definition:

```
def ask_for_int_input(question_string):
```

- And we know what we want the function to do:

```
user_input = int(input(question_string))
```

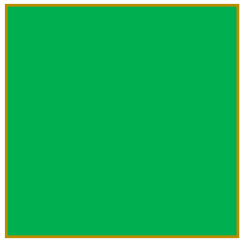
- So, let's put these together:

```
def ask_for_int_input(question_string):
    user_input = int(input(question_string))
```

4 spaces

- Notice the indentation?
  - This is how Python knows what code belongs in what function…
  - Also known as a "block of statements"

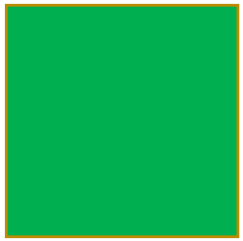
Python Functions



No indenting

```
def ask_for_int_input(question_string):
# save the input from the user in a variable called "user_input"
user_input = int(input(question_string))
# get the first age
input_string_1 = "Please enter age 1: "
age1 = int(input(input_string_1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = int(input(input_string_2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

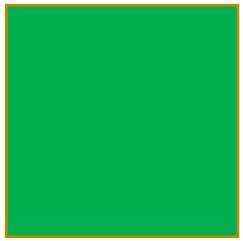
Python Functions



Correct indenting

```
def ask_for_int_input(question_string):
    # save the input from the user in a variable called "user_input"
    user_input = int(input(question_string))

# get the first age
input_string_1 = "Please enter age 1: "
age1 = int(input(input_string_1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = int(input(input_string_2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions



- Functions we have seen so far int() type() input() print()

Python Functions

- Functions we have seen so far int() type() input() print()
- Each of these took a parameter:
  - int(string_to_convert_to_integer)
  - type(parameter_used_to_determine_type)
  - input(parameter_shown_to_user_in_the_form_of_a_question)
  - print(string_printed_to_screen)

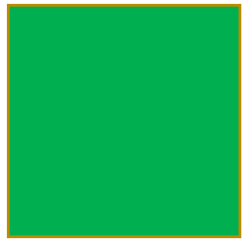
Python Functions

- Functions we have seen so far int() type() input() print()
- Each of these took a parameter:
  - int(string_to_convert_to_integer)
  - type(parameter_used_to_determine_type)
  - input(parameter_shown_to_user_in_the_form_of_a_question)
  - print(string_printed_to_screen)
- Of these, 3 returned a value which we could save in a variable:
  - int(string_to_convert_to_integer) return int
  - type(parameter_used_to_determine_type) return type
  - input(parameter_shown_to_user_in_the_form_of_a_question) return input from user

Python Functions

- Functions we have seen so far int() type() input() print()
- Each of these took a parameter:
  - int(string_to_convert_to_integer)
  - type(parameter_used_to_determine_type)
  - input(parameter_shown_to_user_in_the_form_of_a_question)
  - print(string_printed_to_screen)
- Of these, 3 returned a value which we could save in a variable:
  - int(string_to_convert_to_integer) return int
  - type(parameter_used_to_determine_type) return type
  - input(parameter_shown_to_user_in_the_form_of_a_question) return input from user
- While print() did not return a value

Python reserved keywords



Just known that within Python

these 33 reserved keywords exist

False	def	if	raise
None	del	import	return
True	elif	in	try
and	else	is	while
as	except	lambda	with
assert	finally	nonlocal	yield
break	for	not	
class	from	or	
continue	global	pass	

Python Functions

- So the question to ask yourself is:
- Do we need to return a value from our function?

Python Functions

- So the question to ask yourself is:
- Do we need to return a value from our function?

- In this instance, the answer is yes, as we need the input from the user (keyboard)

Python Functions

- So the question to ask yourself is:
- Do we need to return a value from our function?

- In this instance, the answer is yes, as we need the input from the user (keyboard)

- So, let's add the return statement:

```
def ask_for_int_input(question_string):

    user_input = int(input(question_string))

    return user_input
```

Python Functions

- So the question to ask yourself is:
- Do we need to return a value from our function?

- In this instance, the answer is yes, as we need the input from the user (keyboard)

- So, let's add the return statement:

```
def ask_for_int_input(question_string):

    user_input = int(input(question_string))

    return user_input
```

Python Functions

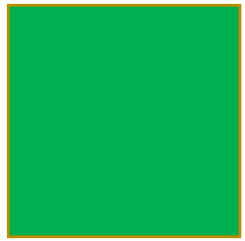
- So the question to ask yourself is:
- Do we need to return a value from our function?

- In this instance, the answer is yes, as we need the input from the user (keyboard)

- So, let's add the return statement and some comments:

```
''' a function that returns an integer value from the user '''
def ask_for_int_input(question_string):
    # save the input from the user in a variable called "user_input"
    user_input = int(input(question_string))
    # return the content of the variable called "user_input"
    return user_input
```

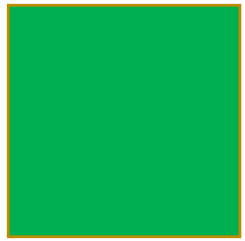
Python Functions



Actual Python Code

```
''' a function that returns an integer value from the user '''


def ask_for_int_input(question_string):
    # save the input from the user in a variable called "user_input"
    user_input = int(input(question_string))
    # return the content of the variable called "user_input"
    return user_input
```

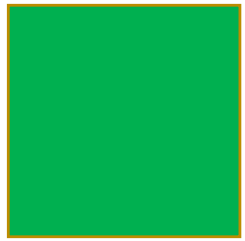
Python Functions



So we've created a function - how do we call it?

```
def ask_for_int_input(question_string):
```

Python Functions



So we've created a function - how do we call it?

```
def ask_for_int_input(question_string):
```

```
age1 = ask_for_int_input("Please enter age 1: ")
```

Python Functions

So we've created a function - how do we call it?

```
def ask_for_int_input(question_string):
```

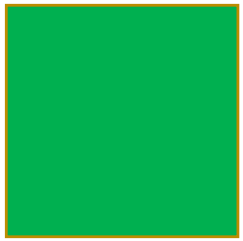
```
age1 = ask_for_int_input("Please enter age 1: ")
```

Returned value

Function call
ask_for_int_input()

Function parameter

Python Functions



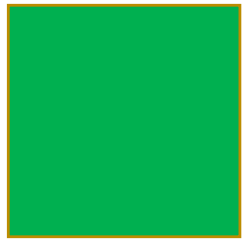
So we've created a function - how do we call it?

```
def ask_for_int_input(question_string):
```

```
age1 = ask_for_int_input("Please enter age 1: ")
```

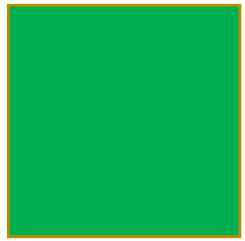
```
age1 = ask_for_int_input("Please enter age 1: ")
# get the second age
age2 = ask_for_int_input("Please enter age 2: ")
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions



```
def ask_for_int_input(question_string):
    # save the input from the user in a variable called "user_input"
    user_input = int(input(question_string))
    # return the content of the variable called "user_input"
    return user_input


# get the first age
input_string_1 = "Please enter age 1: "
age1 = ask_for_int_input(input_string_1)
# get the second age
input_string_2 = "Please enter age 2: "
age2 = ask_for_int_input(input_string_2)
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions

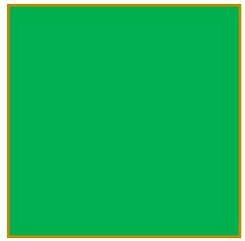


In Python, every variable assignment that is created is given an integer number that uniquely identifies it.

It is guaranteed that no two objects will have the same identifier during any period in which their lifetimes overlap.

To get this value we use the function id()

Python Functions



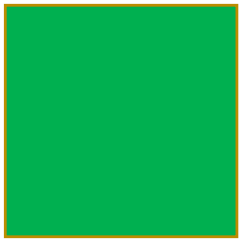
In Python, every variable assignment that is created is given an integer number that uniquely identifies it.

It is guaranteed that no two objects will have the same identifier during any period in which their lifetimes overlap.

To get this value we use the function id()

Not a reserved word, just an inbuilt Python function

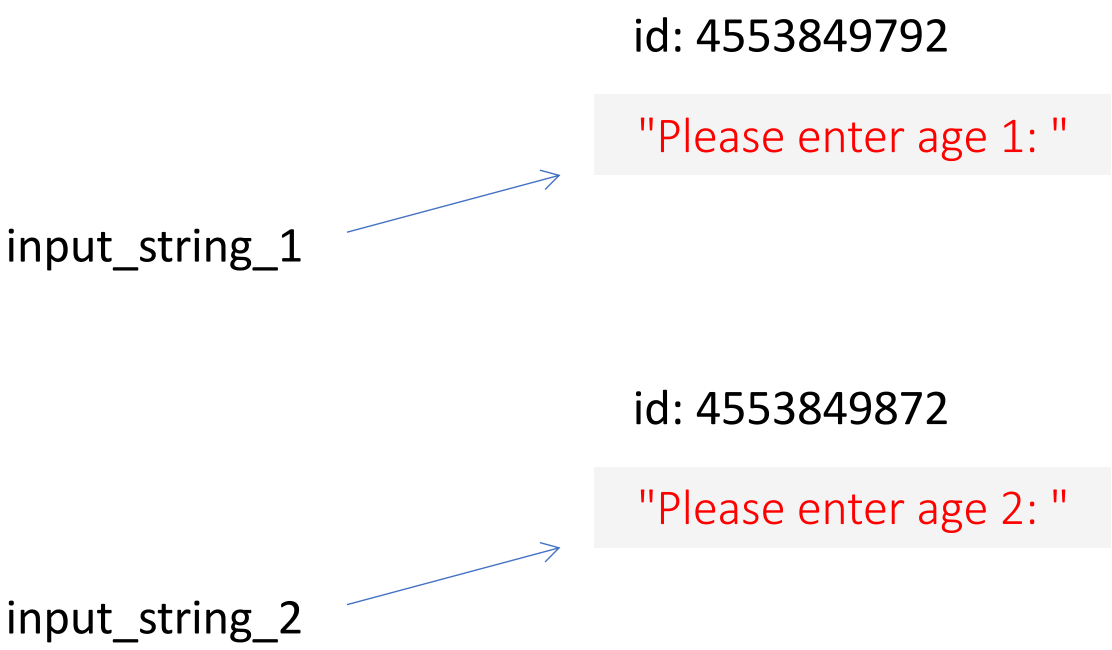
Python Functions



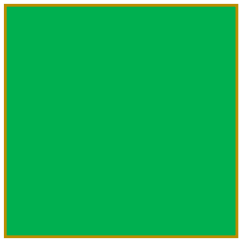
```
# get the first age
input_string_1 = "Please enter age 1: "
print(id(input_string_1))
age1 = ask_for_int_input(input_string_1)
# get the second age
input_string_2 = "Please enter age 2: "
print(id(input_string_2))
age2 = ask_for_int_input(input_string_2)
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
4553849792
Please enter age 1: 2
4553849872
Please enter age 2: 4
The average age is 3
```

Python Functions



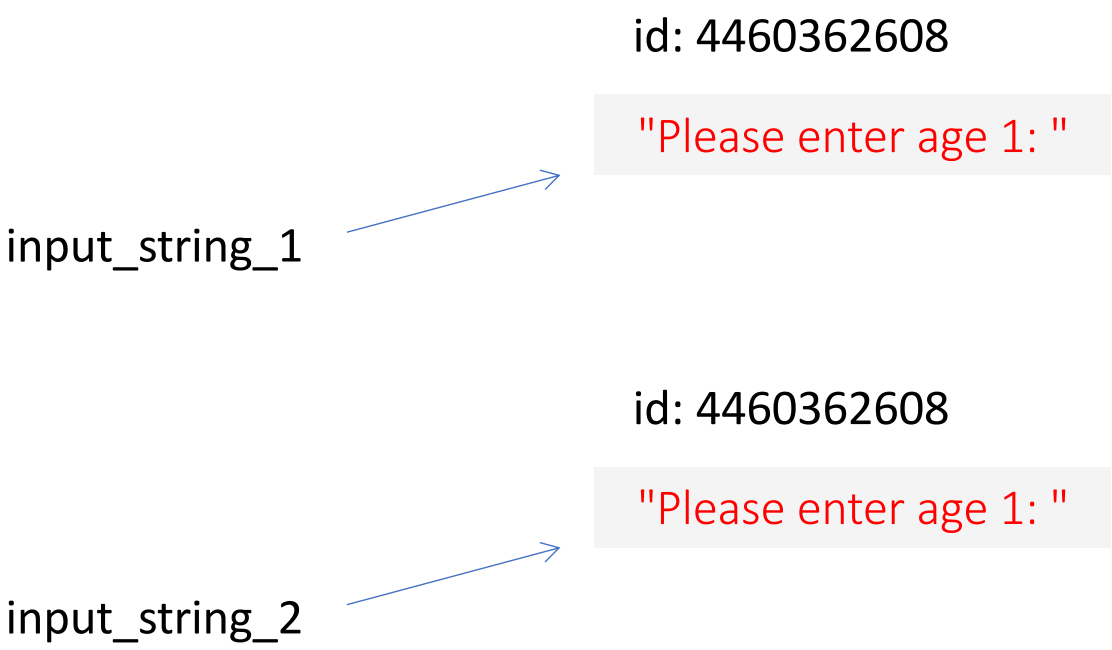
Python Functions



```
# get the first age
input_string_1 = "Please enter age 1: "
print(id(input_string_1))
age1 = ask_for_int_input(input_string_1)
# get the second age
input_string_2 = "Please enter age 1: "
print(id(input_string_2))
age2 = ask_for_int_input(input_string_2)
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

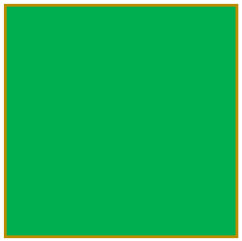
```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
4460362608
Please enter age 1: 2
4460362608
Please enter age 1: 4
The average age is 3
```

Python Functions



Same id – so id is allocated to the item/object, not the variable

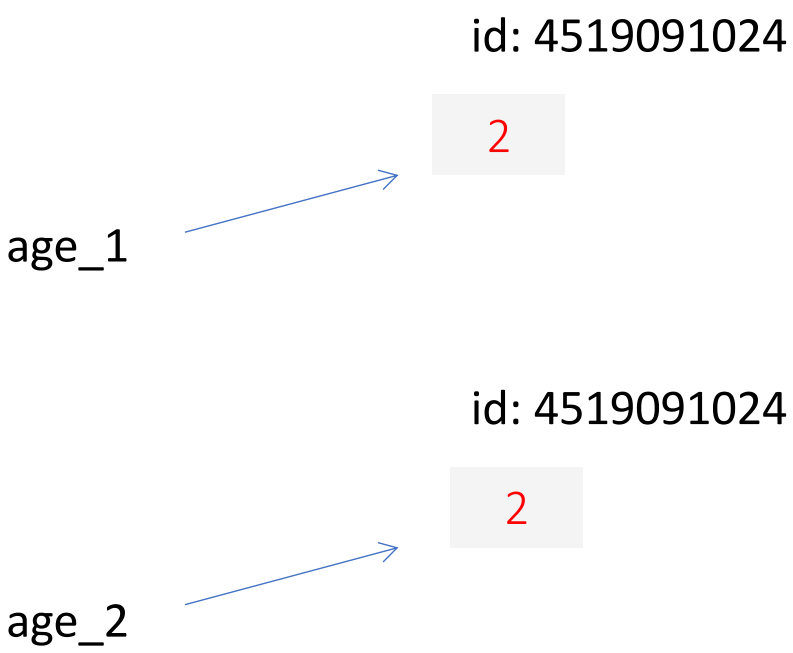
Python Functions



```
# get the first age
input_string_1 = "Please enter age 1: "
age1 = ask_for_int_input(input_string_1)
print(id(age1))
# get the second age
input_string_2 = "Please enter age 2: "
age2 = ask_for_int_input(input_string_2)
print(id(age2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
Please enter age 1: 2
4519091024
Please enter age 2: 2
4519091024
The average age is 2
```

Python Functions



Same id –id is allocated to inputs of the same value also

Python Functions

Change the value of the input and the id changes

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
Please enter age 1: 2
4558551888
Please enter age 2: 4
4558551952
The average age is 3
```

Python Functions

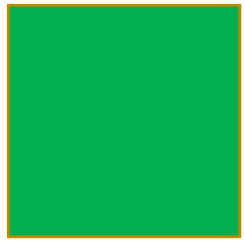
Let’s add some descriptive print statements

```
input_string_1 = "Please enter age 1: "
print("The id for input_string_1 with a value of",
      input_string_1, "is", id(input_string_1))
age1 = ask_for_int_input(input_string_1)
print("The id for age1 with a value of",
      age1, "is", id(age1))
```

Python Functions

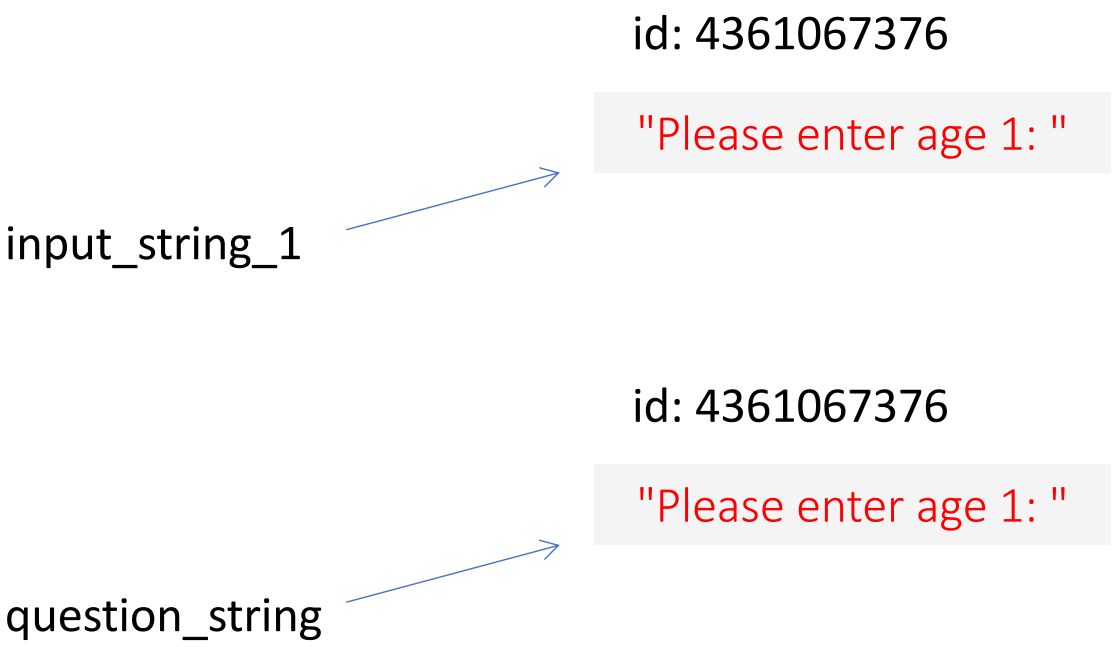
```
def ask_for_int_input(question_string):
    print("The id for question_string with a value of",
          question_string, "is", id(question_string))
    # save the input from the user in a variable called "user_input"
    user_input = int(input(question_string))
    print("The id for user_input with a value of",
          user_input, "is", id(user_input))
    # return the content of the variable called "user_input"
    return user_input


# get the first age
input_string_1 = "Please enter age 1: "
print("The id for input_string_1 with a value of",
      input_string_1, "is", id(input_string_1))
age1 = ask_for_int_input(input_string_1)
print("The id for age1 with a value of",
      age1, "is", id(age1))
# get the second age
input_string_2 = "Please enter age 2: "
print("The id for input_string_2 with a value of",
      input_string_2, "is", id(input_string_2))
age2 = ask_for_int_input(input_string_2)
print("The id for age2 with a value of",
      age2, "is", id(age2))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions



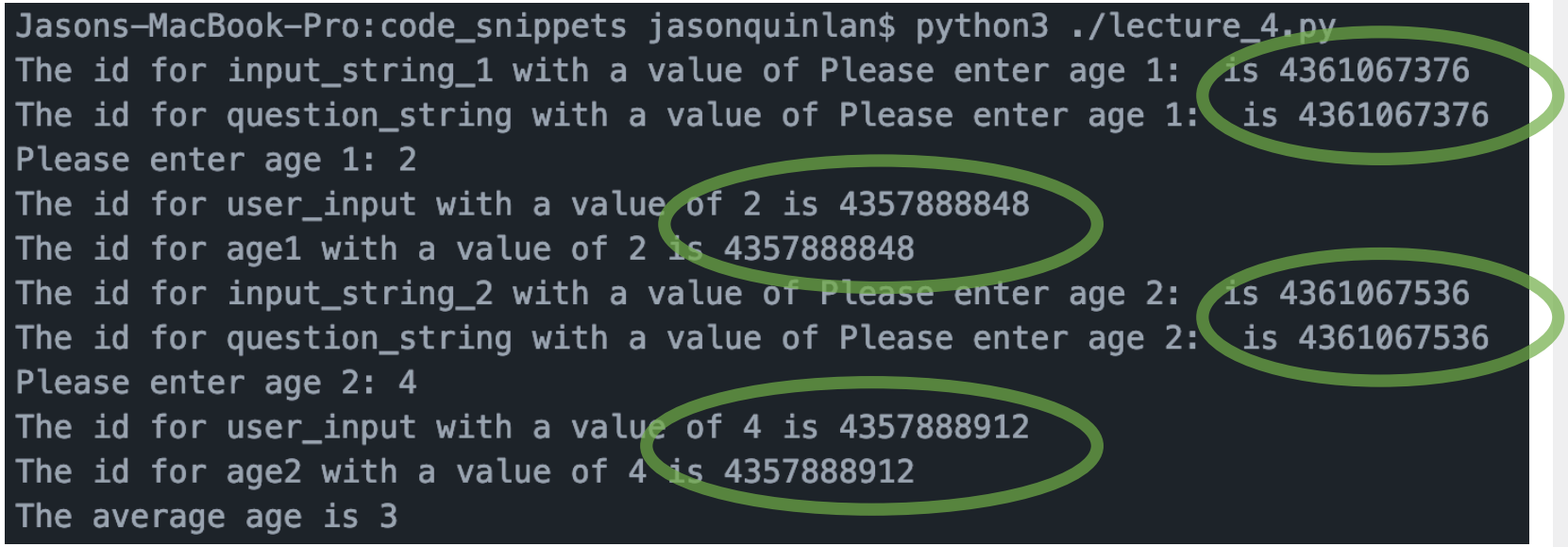
```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
The id for input_string_1 with a value of Please enter age 1:  is 4361067376
The id for question_string with a value of Please enter age 1:  is 4361067376
Please enter age 1: 2
The id for user_input with a value of 2 is 4357888848
The id for age1 with a value of 2 is 4357888848
The id for input_string_2 with a value of Please enter age 2:  is 4361067536
The id for question_string with a value of Please enter age 2:  is 4361067536
Please enter age 2: 4
The id for user_input with a value of 4 is 4357888912
The id for age2 with a value of 4 is 4357888912
The average age is 3
```

Python Functions



So the function parameter is also pointing to the same object as the variable in the function call

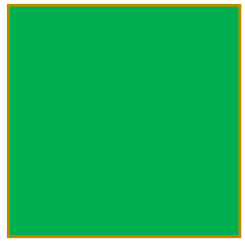
Python Functions



Python Functions Recap

- We saw how we can take similar code and create a function
- We saw how to define the function:
  - def function_name(function_parameter):
- We saw how we indent code within the function
  - Known as a block of statements
  - So Python knows which code belongs in the function
- We saw how to return a value
  - Back to the line of code that called the function
  - And allocate the returning value to a variable
- We used id() to find the unique integer for variable values
  - If variables have the same value, they point to the same object and have the same id
  - Calling a function with a parameter, allocates the same id to the value of both the function parameter and the variable in the function call

Python Functions



Now let’s take a look at the code

Live Code time

Python Functions

Now we can pass and return values, let's look at when we pass more than one parameter:

Let's use the "average = (age1+age2)/2" from our code

We take two values and divide by two

Python Functions

Now we can pass and return values, lets look at when we pass more than one parameter:

Let's use the "average = (age1+age2)/2" from our code

We take two values and divide by two

I want you to take out your laptop

or a piece of paper and define the function

average_of_two()

Python Functions

- Let's go back to our "average_age" code
- And simply take the line of code we need

```
# averge ages:
# get the first age
age1 = int(input("Please enter age 1: "))
# get the second age
age2 = int(input("Please enter age 2: "))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions

```
def average_of_two(age1, age2):
    ''' this is a 'docstring'
    function to determine the average of two ages
    '''
    # determine the average age
    average = (age1+age2)/2
    # return the average age
    return average
```

Age is not the best variable name to use,
as we just want to average a number

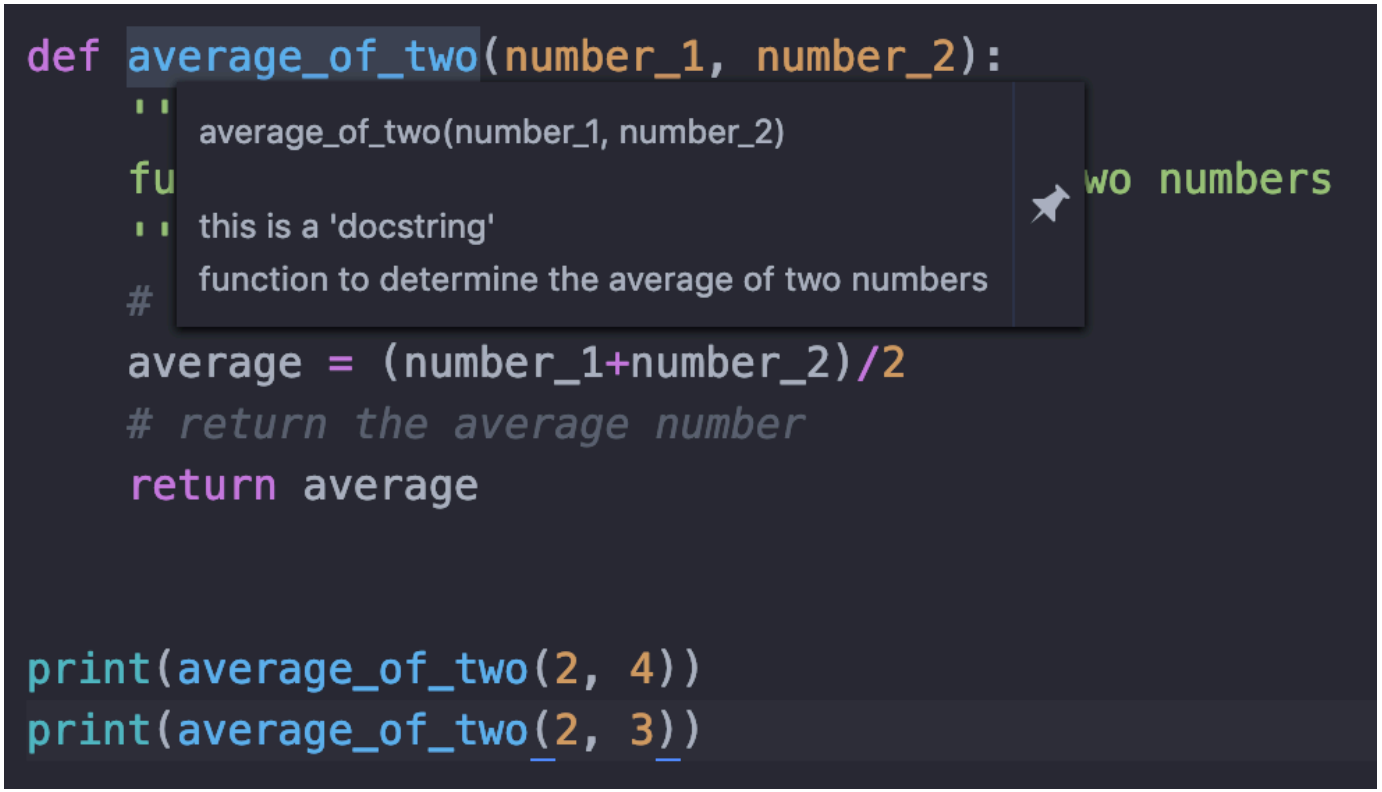
Python Functions

```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average
```

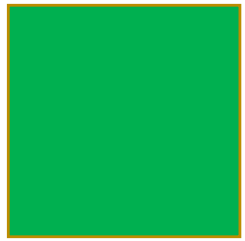
Better:

Note: the docstring comment

Python Functions



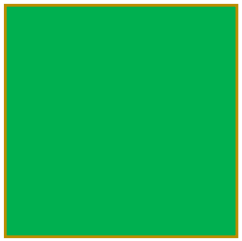
Python Functions



```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))
```

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
3.0
2.5
```

Python Functions



```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))
```

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
3.0
2.5
```

Python Functions

The reason : string formatting

```
# averge ages:
# get the first age
age1 = int(input("Please enter age 1: "))
# get the second age
age2 = int(input("Please enter age 2: "))
# determine the average age
average = (age1+age2)/2
# print to screen
print("The average age is %d" % average)
```

Python Functions

The reason : string formatting

```
print("%d" % average_of_two(2, 4))
```

Will return

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
3.0
2.5
3
```

Python Functions

What about:

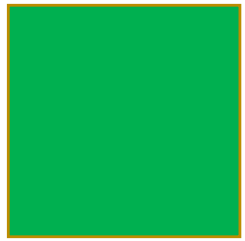
```
print("%d" % average_of_two(2, 3))
```

Will return

```
Jasons-MacBook-Pro:code_snippets jasonquinlan$ python3 ./lecture_4.py
3.0
2.5
3
2
```

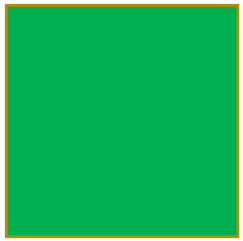
not 2.5 ?? Why?

Python Functions



```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))

print("%d" % average_of_two(2, 4))
print("%d" % average_of_two(2, 3))
```

Python Operators



We have see:

assign (=), addition (+) and float division (/)

Other Python operators include:

multiply (*), subtract (-), exponent (**)

modulus (%), integer division (//)

Python Operators

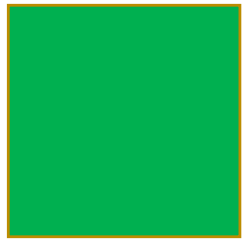
Note: Operators are mutable – they change their operation depending on the data type they're working with.

We have seen this with (+)

String concatenation "Hello " + "World" –> "Hello World"

Addition 2 + 3 -> 5

Python Operators

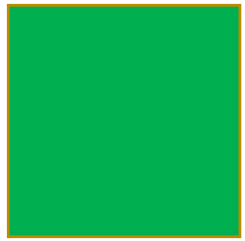


Another good example of this mutability is with multiply (*):

3 * 5 -> 15

print(”Beetlejuice ” * 3) -> "Beetlejuice Beetlejuice Beetlejuice "

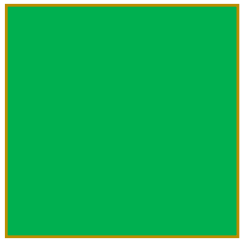
Python Operators



- Float division: / produces a float (13/6 produces 2.1666)
- Integer division: // produces an integer (13//6 produces 2)
- You may need to cast a value to change the data type to get the desired output.

```
x = 13
y = 6
z = x // y
print (z)              #Output is 2
z = float(z)
print (z)              #Output is 2.0
```

Python Operators



The % operator is called the *modulus*.

It will tell you the **remainder** after integer division.

```
x = 11 % 3          #output is 2
x = 10 % 2          #output is 0
```

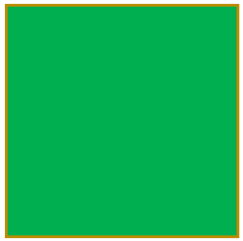
We can use // and % to calculate the answer and remainder:

```
print(10 // 3)      #output is 3
print(10 % 3)       #output is 1
```

The exponent (power of) operator is **

```
x = 2 ** 3          #output is 8
```

Python Operators



Remember BOMDAS?!

Operation enclosed in parentheses are performed first

```
x = (11 - 2) * 3        #output is 27
x = 11 - (2 * 3)        #output is 5
```

If no parentheses are given, order of precedence takes over

- Higher order equations performed first
- Equal order operators applied left -> right

What is the answer for the following?

```
x = 11 - 2 ** 2  + 3 * 2
```

Python Operators

Python's Order of Precedence for operators:

- Exponential **
- Multiplication and Both Divisions and Remainder * / // %
- Addition and Subtraction + -

Now let's look at the Example

```
x = 11 - 2 ** 2  + 3 *2
```

becomes `11 - 4 + 3 * 2`

becomes `11 - 4 + 6`

becomes `7 + 6`

becomes `13`

To avoid unexpected outcomes use parentheses:

```
x = (11 - (2 ** 2))  + (3 *2)
```

Python Functions

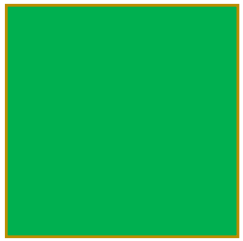
```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))

print("%d" % average_of_two(2, 4))
print("%d" % average_of_two(2, 3))
```

Python Functions

```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))

print("%d" % average_of_two(2, 4))
print("%d" % average_of_two(2, 3))
```

```
Jasons
3.0
2.5
3
2
```

Python Functions



```
def average_of_two(number_1, number_2):
    ''' this is a 'docstring'
    function to determine the average of two numbers
    '''
    # determine the average of two numbers
    average = (number_1+number_2)/2
    # return the average number
    return average


print(average_of_two(2, 4))
print(average_of_two(2, 3))

print("%d" % average_of_two(2, 4))
print("%d" % average_of_two(2, 3))
```

```
Jasons
3.0
2.5
3
2
```

Float division give us a float - > %d in print gives us back an int

Python Functions Recap

- We wrote an “average_of_two” function
  - That calculates the average of two numbers
  - And returns said average
- We add a docstring comment, which is viewable in an IDE
- We noted that our returned value was a float
  - Generating a value with a decimal place
- We looked at Python Operators
  - (=), (+), (/), (*), (-), (**), (%) and (//)
  - We noted that the operators are mutable
    - Change their operation based on the data type they are working with
  - They have precedence (similar to BOMDAS)

UCC
University College Cork, Ireland
Coláiste na hOllscoile Corcaigh