INFT1004

Introduction to Programming

Module 1.1
Introduction to INFT1004

My Contact Details

Course coordinator / Lecturer / Tutor

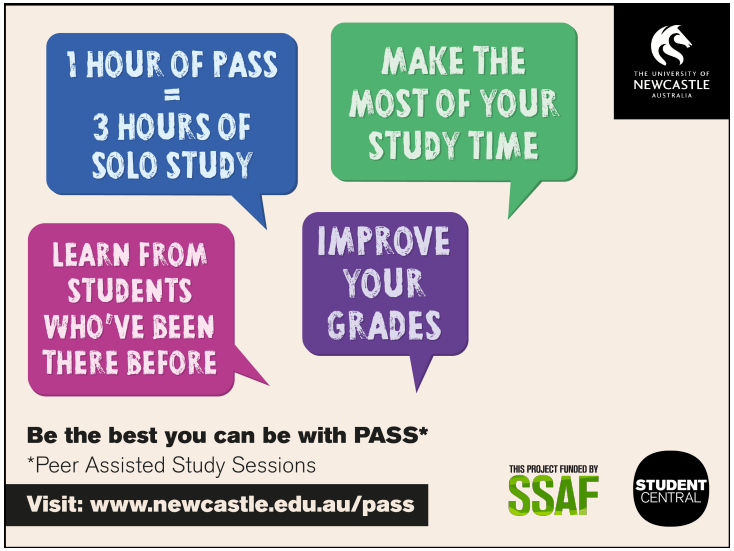
Keith Nesbitt
keith.nesbitt@newcastle.edu.au
ICT3.20
www.knesbitt.com

Research Interests	Perception and cognition Visualisation / Sonification … Computer games Virtual reality Simulation and Training Algorithms and Patterns Creativity

i3 Lab

interaction | interfaces | invention

http://i3lab.newcastle.edu.au/projects.html



GET AHEAD GET ONLINE

Online PASS

The Online PASS pages have been put together by PASS leaders who have studied and excelled in your course before.

You will have access to weekly tips and activities provided by the PASS leader and a discussion board where you can chat with other students.

THIS PROJECT FUNDED BY SSAF

Your PASS leader for INFT1004 is Sharlene Von Drehnen

room change W238

TUESDAY	2pm – 3pm	~~MCG25~~
WEDNESDAY	1pm – 2pm	MCG29

THIS PROJECT FUNDED BY SSAF

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
9:00 - 10:00					
10:00 - 11:00			Consultation ICT3.20	INFT1004 Lab 4 *ICT3.44* Will	
11:00 - 12:00			INFT1004 Lab 1 - BYOD *ICT3.29* Keith		
12:00 - 1:00				INFT1004 Lab 5 *ICT3.44* Will	
1:00 - 2:00			PASS MCG 29		
2:00 - 3:00		PASS W 238	INFT1004 Lab 2 *ICT3.37* Brendan	INFT1004 Lab 5 *ICT3.44* Will	
3:00 - 4:00		INFT1004 Lecture GP 201			
4:00 - 5:00			INFT1004 Lab3 *ICT3.44* Brendan	INFT1004 Lab 6 *ICT3.44* Will	
5:00 - 6:00					
6:00 - 7:00					
7:00 - 8:00					

	MONDAY	TUESDAY	WEDNESDAY	THURSDAY	FRIDAY
9:00 - 10:00					
10:00 - 11:00			Consultation ICT3.20	INFT1004 Lab 4 *ICT3.44* Will	
11:00 - 12:00			INFT1004 Lab 1 - BYOD *ICT3.29* Keith		
12:00 - 1:00				INFT1004 Lab 5 *ICT3.44* Will	
1:00 - 2:00			PASS MCG 29		
2:00 - 3:00			INFT1004 Lab 2 *ICT3.37* Brendan	INFT1004 Lab 5 *ICT3.44* Will	
3:00 - 4:00					
4:00 - 5:00			INFT1004 Lab3 *ICT3.44* Brendan	INFT1004 Lab 6 *ICT3.44* Will	
5:00 - 6:00					
6:00 - 7:00					
7:00 - 8:00			PASS Online		

This is a BYOD – bring your own device lab – so you need to bring a laptop - if you have no laptop with JES then you should go to another lab ☺

Tutorial Times (Don't Panic!)

There are limited places in tutorials (25 computers) and limited seating.

With ~170 students in the course it is not always possible for us to provide people with their preference (apologies).

Don't panic - this becomes less of a problem as the course progresses and numbers tend to reduce.

Personally I am happy for you to come to whatever tutorial you like (even more than 1).

INFT1004 - SEMESTER 1 - 2017		LECTURE TOPICS	
Week 1	Feb 27	Introduction, Assignment, Arithmetic	
Week 2	Mar 6	Sequence, Quick Start, Programming Style	
Week 3	Mar 13	Pictures, Functions, Media Paths	
Week 4	Mar 20	Arrays, Pixels, For Loop, Reference Passing	
Week 5	Mar 27	Nested Loops, Selection, Advanced Pictures	Practical Test
Week 6	Apr 3	Lists, Strings, Input & Output, Files	
Week 7	Apr 10	Drawing Pictures, Program Design, While Loop	Assignment set
Recess	**Apr 14 – Apr 23**	**Mid Semester Recess Break**	
Week 8	Apr 24	No Lecture / Revision and Assignment in Labs	
Week 9	May 1	Data Structures, Processing sound	
Week 10	May 8	Advanced sound	Assignment part 1 due 8:00am Tue, May 9
Week 11	May 15	Movies, Scope, Import	
Week 12	May 22	Turtles, Writing Classes	Assignment part 2 due 8:00am Tue, May 23
Week 13	May 29	Revision	
Mid Year Examination Period - MUST be available normal & supplementary period			

Lecture Topics and Lab topics are the same for each week

Your Contact Details

The University has given you a studentmail address.

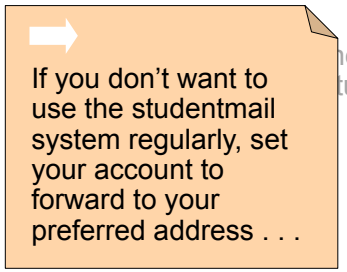
Use it. If the Uni wants to contact you, it does so via this address.

Technically, Uni staff are not permitted to reply to email apparently from a student, but from some other address.

Your Contact Details

The University has given you a studentmail address.

Use it. If the Uni wants to contact you, it does so via this address.



Your Contact Details

The University has given you a studentmail address.

Use it. If the Uni wants to contact you, it does so via this address.

Technically, Uni staff are n
email apparently from a stu
other address.

. . . and remember to log in and delete mail now and then, so that your mailbox doesn't fill up and reject any further incoming messages.

Assessment

Program Quizzes
(most weeks in tutorials)
formative - worth 0%

Practical Test
(held week 5 in your tutorial)
formative - worth 0%

Formative	
Quizzes	0%
Practical Test	0%

These are formative assessment items

(To help you understand how well you are going in the course.)

Feedback will be automatic for quizzes – and self marking for practical test

Assessment

Program Quizzes
(most weeks in tutorials)
formative - worth 0%

Practical Test
(held week 5 in your tutorial)
formative - worth 0%

Programming Assignment

Part 1
(due in week 10) - worth 20%

Part 2
(due in week 12) - worth 30%

Formative	
Quizzes	0%
Practical Test	0%

Summative	
Programming Assignment (part 1)	20%
Programming Assignment (part 2)	30%
	50%

Assessment

Program Quizzes
(most weeks in tutorials)
formative - worth 0%

Practical Test
(held week 5 in your tutorial)
formative - worth 0%

Programming Assignment

Part 1
(due in week 10) - worth 20%

Part 2
(due in week 12) - worth 30%

Final exam
worth 50%
make sure you are available for the whole of the exam period (normal and supplementary)

Formative	
Quizzes	0%
Practical Test	0%

Summative	
Programming Assignment (part 1)	20%
Programming Assignment (part 2)	30%
Final Exam	50%

Assessment

Students must attain 50% overall to pass the course

Programming Assignment

Part 1
(due in week 7) - worth 20%

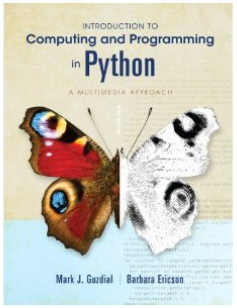
Part 2
(due in week 10) - worth 30%

Final exam
worth 50%
make sure you are available for the whole of the exam period (normal and supplementary)

Formative	
Quizzes	0%
Practical Test	0%

Summative	
Programming Assignment (part 1)	20%
Programming Assignment (part 2)	30%
Final Exam	50%

Old Text Book



Introduction to Computing and Programming with Python
(Third Edition)

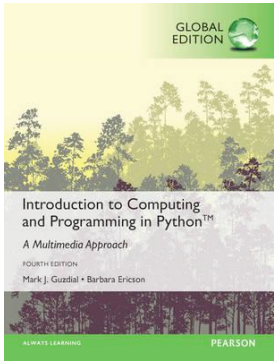
Mark Guzdial & Barbara Ericson

(Pearson, 2013)

This is the text for previous offerings of this course.

Text refers to JES 4.3

New Text Book



Introduction to Computing and Programming with Python
(Fourth Edition – Global Edition)

Mark Guzdial & Barbara Ericson

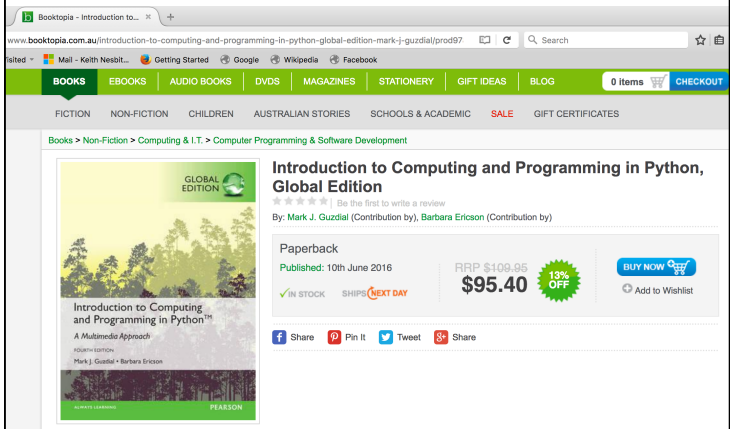
(Pearson, 2016)

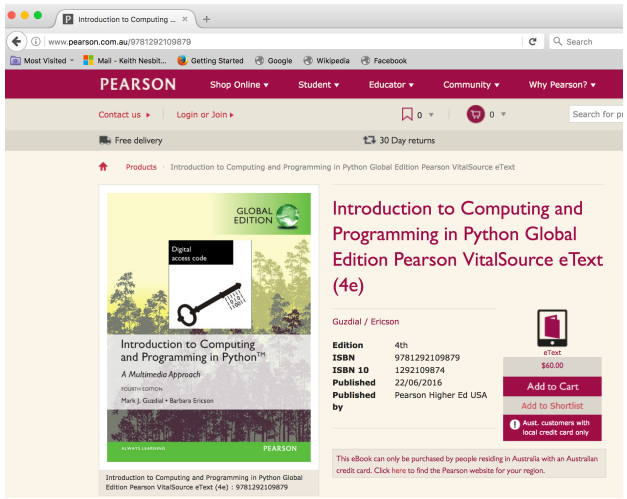
Offers explanatory text and worked examples.

This is the text you want (it is invaluable!)

Text refers to JES 5.02

New Text Book





Myth Busters #1

I don't need a Textbook

The lectures will sometimes use completely different examples, to offer you alternative explanations of the topic.

Therefore you should *both* attend the lectures *and* labs and read and work through the book.

Myth Busters #2

I am a Teacher

No - I am a Lecturer ☺

If you look in a mirror you will see the teacher.

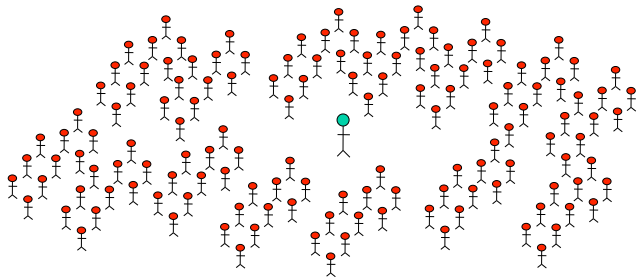
The textbook is very handy to help this process.

Myth Busters #3

Email works well

Myth Busters #3

Email works well



Myth Busters #3

Email works well

This is not an online course – although many materials will be available on Blackboard.

It is assumed that you are attending lectures and labs.

Thanks for seeing me during these times.

This is not an online course

The course runs in face to face mode so it does not attempt to provide an online remote study option.

We do our best to help but also have only limited resources - you should plan to attend lectures and labs.

Sorry but tutors etc are not expected to provide 24/7 email advice or feedback. Please attend class.

Thanks.

Blackboard

Note - lectures and lab information is here.

It contains weekly lab notes and exercises.

It contains lecture slides (made of modules)

It also contains module slides.

You will submit assignments through blackboard.

It contains an electronic copy of the course outline

Course outline

Note the bits about handing in work late.

Note the bits about what to do if your work is affected by illness or other acceptable adverse circumstances. (You will follow the same process for all summative assessment items)

The lecture topics in the course outline are indicative only and may vary – so you should check the course Blackboard website weekly for course updates, bulletins, and additional resources

Rest of course outline

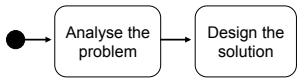
Note what it says about plagiarism, which the Uni (and I) take very seriously.

Note the different dates for withdrawing from the course, and think about what they might mean to you.

INFT1004 Visual Programming

This course will help you learn how to

- comprehend a programming problem and design a solution algorithm
- code the solution algorithm in a specific programming language (Python)
- test and document your program solutions according to suitable standards.



INFT1004 Visual Programming

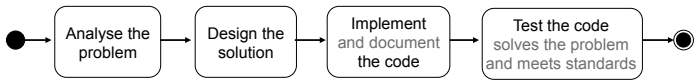
This course will help you learn how to

- comprehend a programming problem and design a solution algorithm
- code the solution algorithm in a specific programming language (Python)
- test and document your program solutions according to suitable standards.

INFT1004 Visual Programming

This course will help you learn how to

- comprehend a programming problem and design a solution algorithm
- code the solution algorithm in a specific programming language (Python)
- test and document your program solutions according to suitable standards.

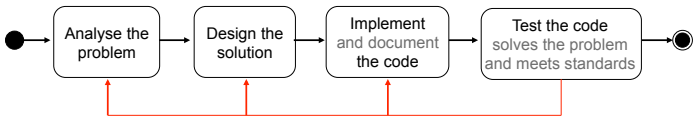


What is Programming

- Programming is deciding on a set of instructions that will get the program to do the required job, then writing those instructions in the form required by the computer (recipe).

What does Programming Involve

- Programming is deciding on a set of instructions that will get the program to do the required job, then writing those instructions in the form required by the computer.
- It often also involves a debugging process intended to eliminate problems in the program (which can sometimes be a frustrating and lengthy experience).



Programming Takes Practice

- Programming is deciding on a set of instructions that will get the program to do the required job, then writing those instructions in the form required by the computer.
- It often also involves a debugging process intended to eliminate problems in the program (which can sometimes be a frustrating and lengthy experience).
- You cannot learn to program by reading and watching; it takes incredible amounts of practice.

Programming

- Programming is deciding on a set of instructions that will get the program to do the required job, then writing those instructions in the form required by the computer.
- It often also involves a debugging process intended to eliminate problems in the program (which can sometimes be a frustrating and lengthy experience).
- You cannot learn to program by reading and watching; it takes incredible amounts of practice.
- Programming courses may take more time than other courses (it varies between people).

programming is something that you need to practice!!

How to Surf ?

1. Paddle really fast when a wave is coming.
2. Stand up when you have caught the wave.
3. Turn a bit, get tubed, do some air etc.
4. If the wave closes out then jump off!

How to Surf ?

Who feels like they can surf now?

Well programming is just like this

I can tell you how to do it and it might even seem pretty easy for some people

Many people find this hard without practice

- YOU NEED TO PRACTICE!

programming is something that you need to practice!!

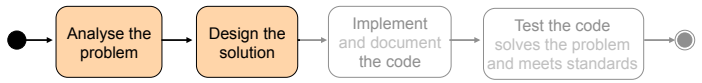
Problem Solving

This course will help you learn how to

- comprehend a programming problem and design a solution algorithm

The first of these is generally the hardest; some people find it quite difficult.

Let's give it a try right now!



Tools to solve problems

Programming languages provide helpful "tools" that allow you to solve a wide range of problems.

These basic tools (program constructs) are:

Sequence (an order of instructions)

Selection (choosing or ignoring some instructions)

Iteration (repeating instructions)

Problem solving

Making a cup of tea or coffee

What are the steps involved? (sequence)

Problem solving

Making a cup of tea or coffee

What are the steps involved? (sequence)

Are there decisions to make? (selection)

Problem solving

Making a cup of tea or coffee

What are the steps involved? (sequence)

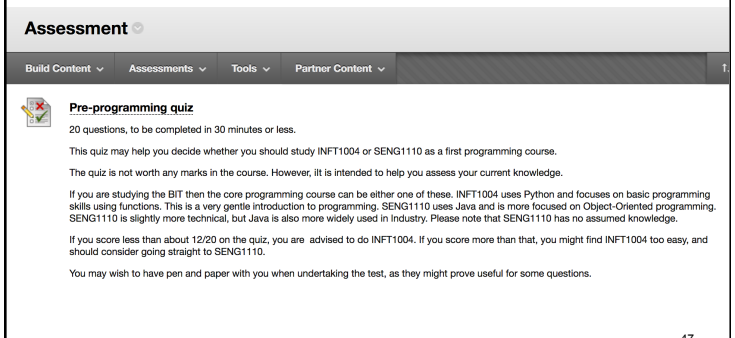
Are there decisions to make? (selection)

What if you work in a coffee shop and need to make 10 cups of tea/coffee? (iteration)

INFT1004 or SENG1110

If you are doing the BIT - either can be your core programming

Not sure - this quiz on blackboard might help you decide



What to do today

- ☐ Buy Guzdial & Ericson textbook:
- ☐ You own the textbook – now start reading it (try chapter 1 & 2)
- ☐ Get JES5.02 installed on your computer (see Module 1.2)
- ☐ You have installed JES – now start practicing!

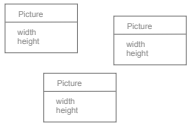
INFT1004

Introduction to Programming

Module 1.2
Introduction to JES
(Hello World)

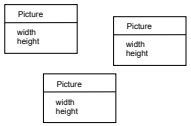
Traditional or object-oriented

Traditional programming	Object-oriented programming
a program is a long sequence of instructions, often involving choice and repetition. Based on functions.	consists of lots of objects (things), each of which can have its own little sequences of instructions.



Traditional or object-oriented

Traditional programming	Object-oriented programming
a program is a long sequence of instructions, often involving choice and repetition. Based on functions.	consists of lots of objects (things), each of which can have its own little sequences of instructions.



Python is object-oriented

Python is object-oriented. But it also allows for functional programming – we will see a mix of approaches)

Object-oriented programming
consists of lots of objects (things), each of which can have its own little sequences of instructions.

(We will be writing a lot of functions)
(We will be dealing with objects quite soon)
(But it will be a while before we write classes)

Python, Jython, JES

There are many, many programming languages

A

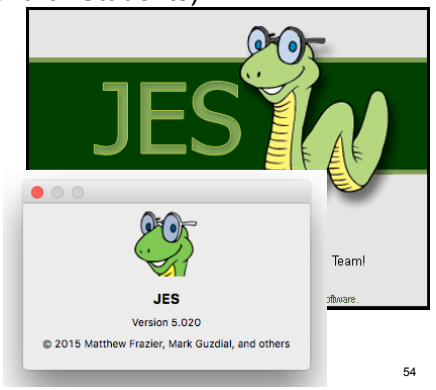
- A# .NET
- A# (Axiom)
- A-0 System
- A+
- A++
- ABAP
- ABC
- ABC ALGOL
- ABLE
- ABSET
- ABSYS
- Abundance
- ACC
- Accent
- Ace DASL
- ACT-III
- Action!
- ActionScript
- Ada
- Adenine
- Agda
- Agora
- AIMMS
- Alef
- ALF
- ALGOL 58
- ALGOL 60
- ALGOL 68
- Alice
- Alma-0
- AmbientTalk
- Amiga E
- AMOS
- AMPL
- APL
- AppleScript
- Arc
- Arden Syntax[1]
- ARexx
- Argus
- AspectJ
- Assembly language
- ATS
- Ateji PX
- AutoHotkey
- Autocoder
- AutoIt
- AutoLISP / Visual LISP
- Averest
- AWK
- Axum

Some are easy to learn; others are widely used professionally; others are for very specific purposes . .

Python, Jython, JES

We're using it in an environment called JES v5.02 (Jython Environment for Students)

which includes lots of features for programming with pictures and sounds



Python, Jython, JES

We're going to use Python, a language that is easier than some to learn, and is used fairly widely, especially with web applications

The version we're using is called Jython – It is Python written in another language called Java

(Python is typically written in C – another language)

Downloads - JES

You can download JES (ver 5.02) and related bits and pieces from the Media Computation site at Georgia Tech, where the textbook's authors work

http://coweb.cc.gatech.edu/mediaComp-teach#Python

It's very easy to install (Windows or Mac)

Note: Ver 5.02 goes with the Fourth (Global) edition of the textbook.

Downloads - JES

http://coweb.cc.gatech.edu/mediaComp-teach#Python

Resources that all students and teachers need:
Latest version of JES Stable (v5.02)
Or download one of these (you will only need one of these):

- JES 5.02 Windows with Installer (For most people, this is what you want.)
- JES 5.02 Windows ZIP if you ALREADY have a JRE
- JES 5.02 Windows ZIP which INCLUDES a JRE
- JES 5.02 Mac OS X Zip
- JES 5.02 Linux Zip

NOTE: JES runs on top of Java – you may already have Java on your machine and want a different install of JES 5.02

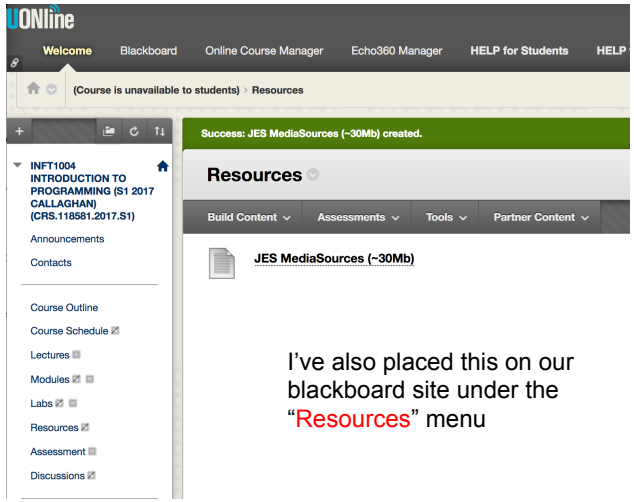
Downloads – MediaSources

You'll also want a copy of MediaSources, a folder full of sounds and image files that are used in the textbook (and course) as examples.

http://coweb.cc.gatech.edu/mediaComp-teach#Python

- 4th Edition Media Sources (free images and sounds, as used in the book): mediasources-4ed.zip
- Python 4ed Errata

Download a copy of mediasources-py4ed.zip and unzip it.



I've also placed this on our blackboard site under the "Resources" menu

JES 4.3 versus JES 5.02

There is not a lot of changes that impact on the materials in the course (So you could use ver 4.3)

https://github.com/gatech-csl/jes/releases

- Upgrading the Jython interpreter to version 2.5, making available new language features and speeding up many user programs.
- Adding code to JES and the installers to support double-clicking .py files to open them in JES, on all supported platforms.
- Bundling JMusic and the Jython Music libraries, allowing JES to be used with the text "Making Music with Computers" by Bill Manaris.
- Adding a plugin system that allows developers to easily bundle libraries for use with JES.
- Fixing the Watcher, so that user programs can be executed at arbitrary speeds.
- Adding new color schemes for the Command Window, which allow users to visually see the difference between return values and print output.
- Fixing numerous bugs.
- Correcting the spelling of "Loading Program."

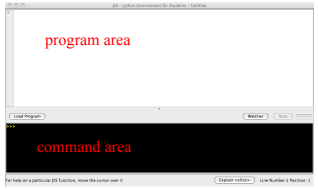
JES Environment

mac version

The JES panes

The top pane of JES is called the program area; that's where we'll write programs (and functions)

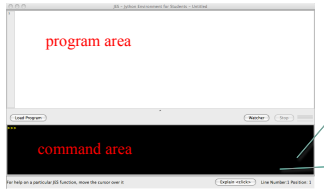
The bottom pane is called the command area; that's where we can type individual commands for JES to execute immediately – programs (and functions)



The JES panes

I usually write programs made up of functions and then call these from the command area.

But the command area is also a good place to practice – you can call simple functions and see what they do – BUT don't try and write functions here!

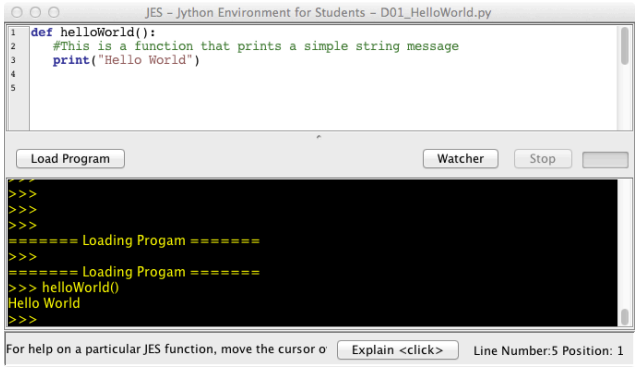


This is also called the interactions area

Don't write programs here!

Hello World

Start JES and create your first program!
This program just contains the `helloWorld` function …



Function Definition

This line declares the function – first you need the keyword "def" (it stands for define).

Then you need to name the function - you make this up – I made up "helloWorld" – seems like a sensible name

```
def helloWorld():
   #This is a function that prints a simple string message
   print("Hello World")
```

JES is very fussy about syntax – even small typos will get it very confused. The error messages will likely seem like gibberish. You really need to be careful and exact – at the beginning you will make lots of errors – but this is a good way to learn – keep it simple – and practice!

Function Definition

This line declares the function – first you need the keyword "def" (it stands for define).

Then you need to name the function - you make this up – I made up "helloWorld" – seems like a sensible name

```
def helloWorld():
   #This is a function that prints a simple string message
   print("Hello World")
```

JES is not so fussy about names as long as things have different names and you don't use names that are already used by JES (keywords – like def).
I am very fussy about names and we will soon come up with rules for naming things! This is an important part of programming style. (This will be marked).

Function Definition

Functions can have parameters (more about these later)
You specify any parameters inside the parantheses () – there are none for this function – so it is just the empty parantheses .

Then finally you need the colon ":" (Don't forget it!)

```
def helloWorld():
   #This is a function that prints a simple string message
   print("Hello World")
```

JES is very fussy about syntax – even small typos will get it very confused. The error messages may seem like gibberish. You really need to be careful and exact – at the beginning you will make lots of errors – but this is a good way to learn – so practice!

Comment

This line is a comment – comments start with the hash character "#" – The computer ignores all comments

```
def helloWorld():
   #This is a function that prints a simple string message
   print("Hello World")
```

While the computer ignores comments your marker will not. I expect sensible comments in your code (we will discuss the types of comments later in the course when we talk about programming style)

Function Definition

This is the line that actually does the work – it is the print function. This function takes one argument (the thing to print). It doesn't actually send anything to the printer it just displays it on the screen. In this case it prints the string "Hello World".

```
def helloWorld():
    #This is a function that prints a simple string message
    print("Hello World")
```

The way the print function works depends a lot on the type of thing being printed. We haven't yet learned about types (but soon). Basically there are simple types and more complex objects and then there are strings which are somewhere in between.

Function Definition

Each line of code (statement) needs to be on a new line

Code needs to be indented to mark the beginning and end of code blocks

```
def helloWorld():
    #This is a function that prints a simple string message
    print("Hello World")
```

I recommend three spaces for indenting! (more programming style). The book suggests two.

You will at some stage forget to indent correctly and JES will get very upset. The first time you do this you will probably have trouble finding the problem – so practice!

Test your function – NOW!

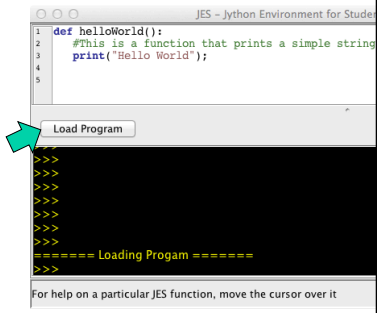
Always test your code in small blocks

I think 1,2,3 (or maybe 4) lines of code is the most you should write before testing.

Test your code actually saves and runs and then test it does what it is meant to do.

Test your function

Before you can actually use the function you will need to save your program and load it.

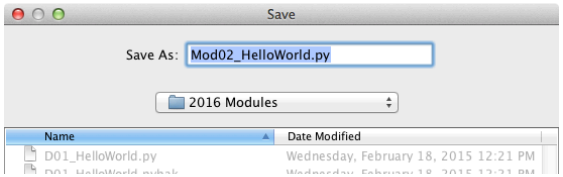


Name your program

When you save your program make sure you give it a sensible name!

Use the ".py" extension for your programs

And save them in a sensible location so you don't loose them!



Hello Name

Extend the first program to contain a second function called `helloName`

```
def helloWorld():
   #This is a function that prints a simple string message
   print("Hello World")

def helloName(name):
   #This is a function that uses a parameter (name) to
   #print a message
   print("Hello " + name)
```


Hello Name

Note that this function has been declared like the first one although it has one parameter called "name"

```
def helloName(name):
   #This is a function that uses a parameter (name) to
   #print a message
   print("Hello " + name)
```


Hello Name

The print statements joins (concatenates) the two strings "Hello " and the name you supply when you call the function before printing.

```
def helloName(name):
   #This is a function that uses a parameter (name) to
   #print a message
   print("Hello " + name)
```

concatenate (join two strings)

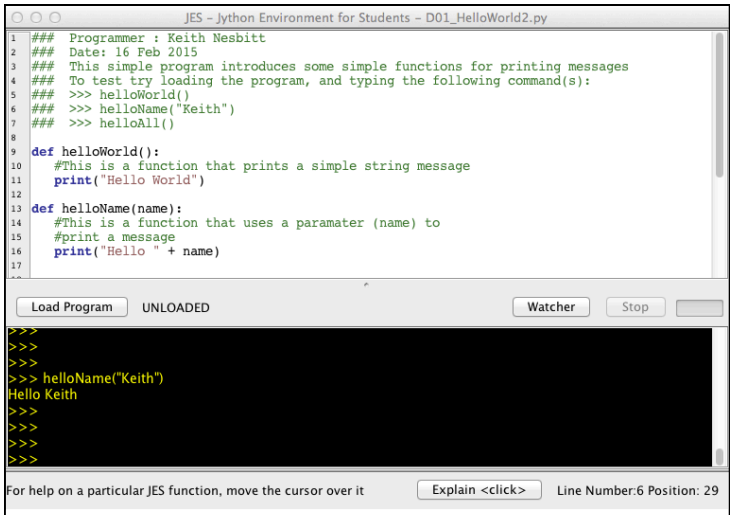
Hello Name

When you call the function make sure you supply an argument (To match the parameter)

```
>>> helloName("Keith")
```

```
def helloName(name):
   #This is a function that uses a parameter (name) to
   #print a message
   print("Hello " + name)
```

concatenate (join two strings)



Mod1_2_HelloWorld.py

Hello All

Extend the first program to contain a third function called `helloAll`

```
def helloAll():
   # This function uses some previously defined functions to
   # print a number of messages
   helloWorld()
   helloName("Keith")
   print ("Hello All!")
```

This function uses some of our previous functions – helloWorld and helloName.

Hello All

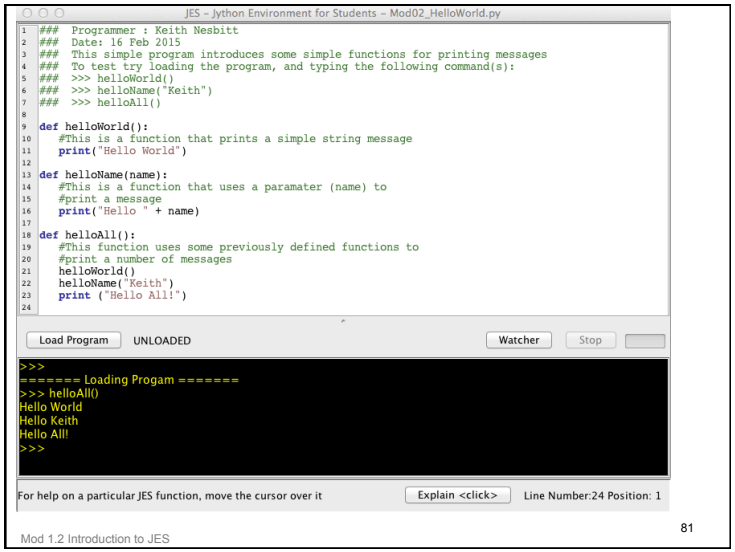
When you call a function make sure you get the number of arguments correct. (In this case none).

No Parameters

```
def helloAll():
   # This function uses some previously defined functions to
   # print a number of messages
   helloWorld()
   helloName("Keith")
   print ("Hello All!")
```

No Arguments

```
>>> helloAll()
```



INFT1004

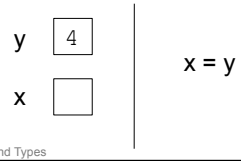
Introduction to Programming

Module 1.3
Assignment and Types

Assignment "="

'x = y' does not mean 'x is equal to y'

It means 'take the value of y (whatever that is) and assign it to the thing called x'

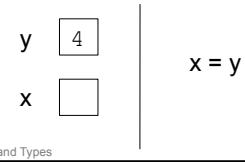


Assignment "="

'x = y' does not mean 'x is equal to y'

It means 'take the value of y (whatever that is) and assign it to the thing called x'

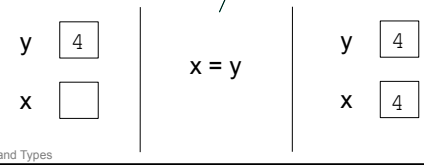
Actually x and y are both ***variables*** – they are names for somewhere in the computers memory that can store data



'=' doesn't mean 'equals'

'x = y' is called an assignment statement

Once this statement has been done, the 'variable' called x will have the same value as y



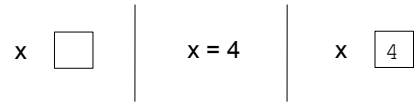
'=' doesn't mean 'equals'

'x = y' is called an assignment statement

The assignment statement takes the value of what's on its right and assigns it to the variable on its left

In this case the thing on the right "y" is also a variable, so the value is taken from this variable "y".

A simpler example is to use a literal on the right eg. x = 4



Literal

This is when a value is written exactly as it is meant to be interpreted.

These are often used to initialise variables (or constants)

e.g.

```
name = "keith"      # string
x = 4               # integer
height = 3.45       # float
```

Constants

Constants are things that keep the same value throughout a program.

```
PI = 3.14159

MAX_HEIGHT = 210

TAX_RATE = 0.45
```

Python doesn't actually allow you to declare constants (Many languages do).

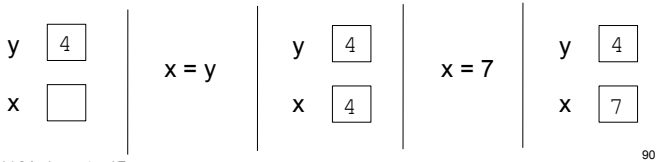
We have to use variables – but we will expect the programmer not to change them.

Variables

Variables are things that *can* change (vary) their value

(So you can put something different in the memory)

(Constants don't change value)



Variables

These are things you will use every so often

You can decide when you want to have one – they are useful for storing information.

(There is no correct time to have or not have a variable – create one when you need to store something.)

Variables

They have a name (which you make up)

It should be a sensible meaningful name (even x and y will be sensible meaningful names in some situations)

You will follow a standard (described in programming style)

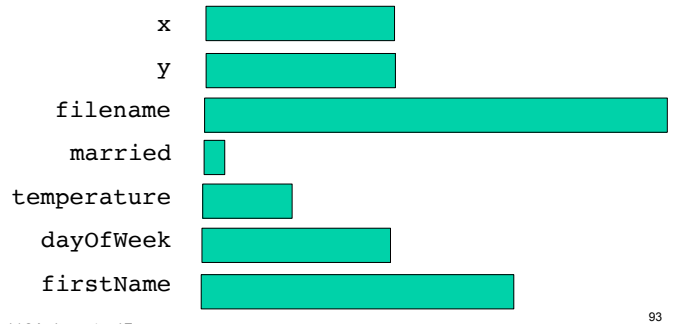
some examples

```
x
y
filename
married
temperature
dayOfWeek
firstName
```

Variables

They provide a label to some memory in the computer where you can store things.

How much memory depends on the type of variable



Variables

In many programming languages you need to declare variables before you use them

When you declare them you also have to say what type they are going to be (they can never change type).

(These are called strongly typed languages)

Types

Python is *clever* enough to work out what the type a variable is - it takes on the type of whatever you last assign to it.

While Python is clever, programmers are not so much (At least not all the time) – this can lead to some problems

If you accidently use the same name – Python won't care it will just overwrite the first one.

So the type of a variable can change during the program (watch out!)

Simple types

integer – a whole number

float – a number with a decimal point (even if the decimal point is followed by zeros)

string – a sequence of characters enclosed in quotation marks

boolean – true/false (actually an integer in python)

Simple types

integer – a whole number

3, 9, 345, 0, -12, 5000

float – a number with a decimal point (even if the decimal point is followed by zeros)

23.40, -233.336, -11.2,

string – a sequence of characters enclosed in quotation marks

"This is an example"

boolean – true/false (actually an integer in python)

0 – false, 1 is true (or any integer > 1)

Simple types

The words true (1) and false (0) are also defined as constants – You should use these words rather than the integer values)

boolean – true/false (actually an integer in python)

0 – false, 1 is true (or any integer > 1)

character – a single character enclosed in quotation marks

"a", "A", "#"

More complicated types

The other types we need to know about will be objects

(Instantiated from a class we write ourselves or a class that JES or Python provides)

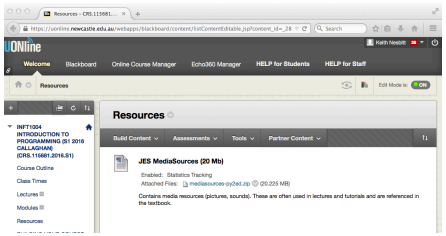
Here are some JES examples of classes

Picture
Sound
File
List

An example - Picture

We will soon learn a lot about the Picture class in JES – We will be using a lot of Picture objects

For now try reading, showing and printing a picture from the MediaSources file that comes with the textbook (on blackboard)



JES and Objects

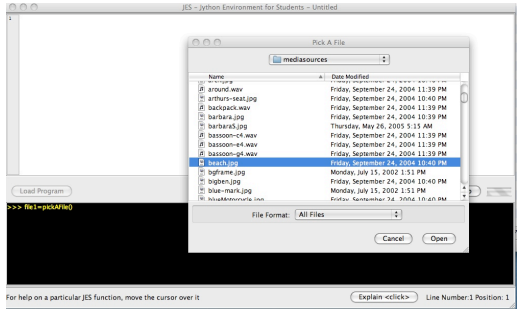
In the JES command window , type

```
>>> pictureFile=pickAFile()
```

and press Enter

JES and Objects

Choose a picture file, eg beach.jpg, from the mediasources folder that goes with the textbook (You can get this from blackboard)



JES and Objects

You now have something called `pictureFile`, and that object is a String

In the JES command window , type

```
>>> print pictureFile()
```

and press Enter

JES and Objects

Use the `pictureFile` to make a **Picture** object

Type

```
>>> beachPicture = makePicture(pictureFile)
```

and press Enter

JES and Objects

You now have another object called `beachPicture`, and that object is a Picture

In the JES command window, type

```
>>> print beachPicture
```

and press Enter

JES and Objects

You now have another object called `beachPicture`, and that object is a Picture

In the JES command window, type

```
>>> print beachPicture
```

and press Enter

(You get some information about the picture)

```
Picture, filename /mediasources/beach.jpg height 480 width 640
```


JES and Objects

Everything in a programming environment has a type: Picture is just one example of more complex types (classes) that we can use to make objects

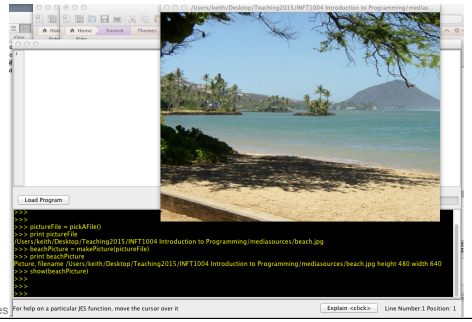
Different types (classes) can have different things done with them – lets try the show function

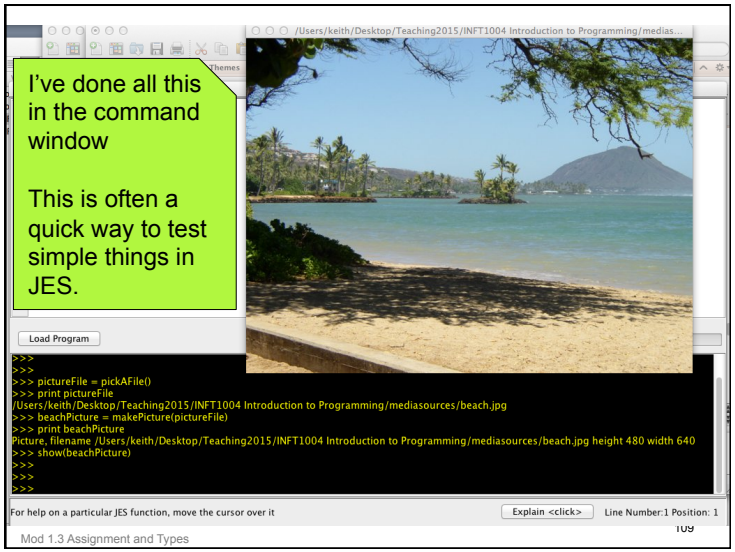
JES and Objects

Type

```
>>> show(beachPicture)
```

and press Enter





Here is a program to do the same thing. This is how you will normally work. Write a program (one or more functions) and call a function from command window.

```
def testPickShowPicture():
  # This function uses pickAFile function to select a picture file
  # it prints the path and name of the the file and then
  # prints and shows the picture
  #
  # Note: There is no testing that you actually get a picture file
  # nor that it is the beach picture.

  pictureFile = pickAFile()
  print pictureFile

  beachPicture = makePicture(pictureFile)
  print beachPicture

  show(beachPicture)
```


Mod1_3_testPrintPictureVariable.py


```
>>> testPickShowPicture()
```


Print

With objects, it doesn't display the object itself; it displays information about the object

```
>>> print beachPicture
```
Picture, filename /mediasources/beach.jpg height 480 width 640

Print

With objects, it doesn't display the object itself; it displays information about the object

With simpler types, like integers, floats, strings, it displays them directly – try...

```
print(13)
print(-8.275)
print("My name")
```


Print

With objects, it doesn't display the object itself; it displays information about the object

With simpler types, like integers, floats, strings, it displays them directly – try...

```
print(13)
print(-8.275)
print("My name")
```

Here we are printing literals - we can try the same thing with variables

Printing some variables

```
luckyNumber = 13
bankBalance = -8.275
message = "My name"


print(luckyNumber)
print(bankBalance)
print(message)
```

Mod1_3_testPrintPictureVariable.py

INFT1004

Introduction to Programming

Module 1.4
Arithmetic

Computers do Arithmetic

Computers were designed to do arithmetic.

As a programmer it is hard to avoid some kind of mathematical thinking in solving problems.

In this course we will keep things rather simple.

But it cannot be avoided completely.

Arithmetic Operators

+	Addition	2 + 3	5
		2.2 + 3.4	5.6

Arithmetic Operators

+	Addition	2 + 3	5
		2.2 + 3.4	5.6
-	Subtraction	3-1	2
		3.4 – 1.1	2.3

Arithmetic Operators

+	Addition	2 + 3	5
		2.2 + 3.4	5.6
-	Subtraction	3-1	2
		3.4 – 1.1	2.3
*	Multiplication	2 * 3	6
		3 * 3.1	9.3

Arithmetic Operators

+	Addition	2 + 3	5
		2.2 + 3.4	5.6
-	Subtraction	3-1	2
		3.4 – 1.1	2.3
*	Multiplication	2 * 3	6
		3 + 3.1	9.3
/	Division	3.0/2	1.5
		3/2	1

Arithmetic Operators

Notice that the result of dividing two integers may not be what you expect

/	Division	3.0/2	1.5
		3/2	1

Some Tricky Arithmetic Operators

%	Modulus	9 % 4	1
		9 % 3	0
		9 % 5	4

You can also think of it as giving the remainder of a division

Some Tricky Arithmetic Operators

%	Modulus	9 % 4	1
		9 % 3	0
		9 % 5	4

This seems to be a very useful function in programming.

```
125 / 60     2
125 % 60     5

  125 minutes
   is 2 hours
and 5 minutes
```

Some Tricky Arithmetic Operators

%	Modulus	9 % 4	1
		9 % 3	0
		9 % 5	4
**	Exponent	3**2	9
		2**3	8
		4**0.5	2
		9**0.5	3

Some Tricky Arithmetic Operators

%	Modulus	9 % 4	1
		9 % 3	0
		9 % 5	4
**	Exponent	3**2	9
		2**3	8
		4**0.5	2
		9**0.5	3

`3**2` is the same as 3 squared (3 to power of 2)

`4**0.5` is the same as taking the square root of 4

Order of Operation

Arithmetic operations are calculated in the order of precedence.

Highest	**			
	*	/	//	%
Lowest	+	-		

If the operations have equal precedence then they are calculated from left to right

Order of Operation

BE careful (use parentheses if you need to)

Highest	**			
	*	/	//	%
Lowest	+	-		

```
>>> 2 + 5 * 8          42
>>> 2 * 5 + 8          18
>>> 2 * (5 + 8)        26
```

Arithmetic in Python

Here are some things to try – and to work at understanding:

```
>>> print(513 * 25)
>>> size = 513 * 25
>>> print(size)
>>> print(3 + 25)
>>> minutes = 60 - 8
>>> print(minutes)

>>> print(2 + 5 * 8)
>>> print(2 * 5 + 8)
>>> print(2 * (5 + 8))

>>> quotient = 13 / 2
>>> print(quotient)
>>> quotient = 13.0 / 2
>>> print(quotient)
```


Commands

Here are some things to try – and to work at understanding:

```
>>> print(513 * 25)
>>> size = 513 * 25
>>> print(size)
>>> print(3 + 25)
>>> minutes = 60 - 8
>>> print(minutes)

>>> print(2 + 5 * 8)
>>> print(2 * 5 + 8)
>>> print(2 * (5 + 8))

>>> quotient = 13 / 2
>>> print(quotient)
>>> quotient = 13.0 / 2
>>> print(quotient)
```

You can try these as single commands (in the bottom window of JES)

Program

```
def testSimpleArithmetic():

    #This function plays with some simple arithmetic
    # using the arithmetic operators (+, -, *, /)
    #it uses the print statement to print the results

    #test some simple arithmetic
    print(513 * 25)
    size = 513 * 25
    print(size)
    print(3 + 25)
    minutes = 60 - 8
    print(minutes)

    #test some order of operation
    print(2 + 5 * 8)
    print(2 * 5 + 8)
    print(2 * (5 + 8))

    #test some division
    quotient = 13 / 2   #result is
    print(quotient)
    quotient = 13.0 / 2 #result is
    print(quotient)
```

You can try putting them into a program as a function (in the top window of JES) and then calling the function as a single command (in the bottom window of JES)

```
>>> testSimpleArithmetic()
```

Mod1_4_testArithmetic.py

Reminders

Division of two integers always gives an integer result

```
13 / 2          6
```

Division of two numbers - when at least one is a float gives a float result

```
13 / 2.0        6.5
```

Don't forget this, or you'll be puzzled now and then!

Test Tricky Arithmetic

```
def testTrickyArithmetic():

   #test some simple arithmetic - modulus
   print(9 % 4)
   print(9 % 3)
   print(9 % 3)

   totalMinutes = 125
   hours = totalMinutes / 60
   minutes = totalMinutes % 60

   #test some simple arithmetic - exponent
   print(3 ** 2) # 3 squared
   print(4 ** 2) # 4 squared
   print(5 ** 2) # 5 squared

   print(2 ** 3) # (2 to power 3) 2 cubed
   print(2 ** 4) # (2 to power 4)
   print(2 ** 5) # (2 to power 5)

   print(4 ** 0.5)   # square root of 4
   print(9 ** 0.5)   # square root of 9
   print(16 ** 0.5)  # square root of 16
   print(25 ** 0.5)  # square root of 25
   print(9.9 ** 0.5) # square root of 9
```


Turn Strings to numbers

Sometimes you might have a string that looks like a number and you want to turn it into a number.

```
myStringInteger = "-9”     #string
myStringFloat =   "47.3”   #string
```


Turn Strings to numbers

Sometimes you might have a string that looks like a number and you want to turn it into a number.

```
myStringInteger = "-9”     #string
myStringFloat =   "47.3”   #string


myInteger = int(myStringInteger) #integer
myFloat = float(myStringFloat)   #float

#be careful try this
myInteger = int(myStringFloat)
```


Turn Strings to numbers

```
def testTurnStringsToNumbers():
   # This function demonstrates how strings can be turned into
   # numbers (useful when reading strings from a file) or
   # anytime you have a string that looks like a number and you
   # want to make it a number!

   myStringInteger = "-9"    # these are strings of characters
   myStringFloat =   "47.3"  # they are not really numbers

   #turn a string into an integer
   myInteger = int(myStringInteger)
   print(myInteger)

   #turn a string into a float
   myFloat = float(myStringFloat)
   print(myFloat)

   #these are really numbers - so try some arithmetic
   print(myInteger + myFloat)
```

Turn numbers to strings

Sometimes you might have a number that you want to turn into a string – For example, I do this a lot when I want to join (concatenate) a string and number together – so I can print a "nice" message.

```
myInteger = 42              # integer
myFloat = 8.6               # float

print("myInteger=" + str(myInteger))
print("myFloat=" + str(myFloat))
```


Turn numbers to strings

```
def testTurnNumbersToStrings():

    # How numbers can be turned into strings
    # (useful for printing numbers)
    myInteger = 42
    myFloat = 8.6

    # print is clever enough to work with integers or
    # floats or strings
    print(myInteger)
    print(myFloat)
    print(myInteger + myFloat)

    # but if you want to concatenate (join) strings and numbers
    # you will need to turn your number into a string first
    # you can do this using the str() command
    print("myInteger=" + str(myInteger))
    print("myFloat=" + str(myFloat))
    print("myFloat=" + str(myInteger + myFloat))
```