Reminders

Australian National University

- Four important To-do items:
  - Fill in the Demographic Information Questionaire on Wattle
  - Sign up to a lab group
    - via myTimetable: https://mytimetable.anu.edu.au/odd/student
  - Login to STREAMS before your first lab (this is required to set things up for you):
    - https://cs.anu.edu.au/streams/login.php
  - Complete the On/Off Campus declaration on course Wattle page
- Student course representative wanted
- Read the news forum on Wattle

Variables (part II)

COMP1730 & COMP6730

Reading: Textbook chapter 2 : Alex Downey, *Think Python*, 2nd Edition (2016)

Australian National University

Revision: every variable has a type

Australian National University

- Variable types in python:
  - Integers (type `int`)
  - Floating-point numbers (type `float`)
  - Text strings (type `str`)
  - Truth or Boolean values (type `bool`)

- Variable types determine what we can do with values (and sometimes what the result is)

- The `type()` function tells us the type of a variable:

```
python                                                    ⌥⌘1
bash-3.2$
bash-3.2$ python
Python 3.9.13 (main, Aug 25 2022, 18:29:29)
[Clang 12.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> type(2)
<class 'int'>
>>> type(2 / 3)
<class 'float'>
>>> type("zero")
<class 'str'>
>>> type("1")
<class 'str'>
>>> type(1 < 0)
<class 'bool'>
>>> type(False)
<class 'bool'>
>>>
>>>
```

Terminal or iTerm programs on a Mac

Type casting: `int()`, `str()`, `float()`

Australian National University

- When we want to convert between variable types, or be *explicit* about type when it may not be obvious -- use `int()`, `str()` and `float()`
- There is no automatic type conversion. So, need to convert between types when necessary.

```
>>>
>>> string_var = 'some text'
>>> int_var = 4
>>> float_var = 4.4
>>> str(float_var)
'4.4'
>>> int(float_var)
4
>>> float(str_var)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'str_var' is not defined
>>>
```

Most variables can be changed

Australian National University

- Mutable == can changed, Immutable == can't be changed
  - All variables you have encountered in Python, thus far, are Mutable
  - There are sometimes good reasons why an Immutable variable is good
- Most variables in your programs can change – and that is good too

```
>>> x = 5
>>> x
5
>>> x = 7
>>> x
7
```

- Increment and Decrement for counting

```
>>> x = 0
>>> x = x + 1
```

Downey (2015) Think Python, 2nd Ed.

Variable assignment

Australian National University

- A variable assignment is written:

```
var_name = expression
```

- This ‘=‘ means something different to this ‘==‘
- When executing an assignment, the interpreter:
  1. Evaulates the right-hand side expression
  2. Associates the left-hand side name with the resulting value

Division means a `float`

Australian National University

- Every constant (literal) with a decimal point represents a `float`:

```
>>> type(1.5 - 0.5)
<class 'float'>
>>> type(1.0)
<class 'float'>
```

- The result of division is always a float:

```
>>> type(4 / 2)
<class 'float'>
```

- Floats an be written (and sometimes printed) in scientific notation:
  - `2.99e8` means $2.99 \cdot 10^{8}$
  - `6.626e-34` means $1.626 \cdot 10^{-34}$
  - `1e308` means $1 \cdot 10^{308}$

String variables are sequences

Australian National University

- Each of the characters in a string may be treated individually. Because `str` variables are **sequences**. More on this in later lectures.
- To access each character in a string, you use the index value (enclosed in square brackets []:

```
>>>
>>> some_text = "Hello, world!"
>>> some_text
'Hello, world!'
>>> some_text[0]
'H'
>>> some_text[5]
','
>>> some_text[7]
'w'
>>>
```

- Index values always start counting from zero!

Lists (quick mention)

Australian National University

- There are other sequence variable types in python. These are very useful and the topic of whole later lectures.
- Lists – a sequence of variables that are ordered by index.

```
>>>
>>> first_list = ["0", 1, 2, "three", 4.0]
>>> first_list
['0', 1, 2, 'three', 4.0]
>>> first_list[0]
'0'
>>> first_list[3]
'three'
>>> first_list[4]
4.0
>>>
```

- Lists may contain variables of mixed types, or of a single type.

Dictionaries (quick mention)

Australian National University

- Abbreviated as `dict` variables. Also the topic of a whole lecture, later.
- For storing **key-value pairs** in a single variable. Can use as a **lookup table**. Also very useful:

```
>>>
>>> first_dictionary = {'DOCK2': 'Dedicator of cytokinesis 2', 'CD4': 'Cluster
 of differentiation 4', 'something_else': 123}
>>>
>>> first_dictionary
{'DOCK2': 'Dedicator of cytokinesis 2', 'CD4': 'Cluster of differentiation 4',
 'something_else': 123}
>>>
>>> first_dictionary['DOCK2']
'Dedicator of cytokinesis 2'
>>>
```

- Note the curly braces '{}' for defining a `dict`, and the square brackets '[]' for accessing the values by key.

Variable names

Australian National University

- There are simple rules that govern the names that can be given to variables.
- Good coding practice: make meaningful names that aid understanding
- Names can be long and contain:
  - Uppercase letters
  - Lowercase letters
  - Numbers
  - Underscores _ but not spaces (`this_is_a_variable`, not `this is a variable`)
- Must not start with a number
- Must not contain symbols or illegal characters (apart from underscore)
- Here are some illegal variable names:

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax
>>> more@ = 1000000
SyntaxError: invalid syntax
>>> class = 'Advanced Theoretical Zymurgy'
SyntaxError: invalid syntax
```

Downey (2015) Think Python, 2nd Ed.

Variable names must not be Python keywords

Australian National University

- Don’t try to use these as variable names:

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

Downey (2015) Think Python, 2nd Ed.

- This will become natural for you after some practice

Avoid built-in function names too

Australian National University

- Will get to built-in functions in a moment. But these names make bad variable names:
- The problem is that these names will ‘work’ – but have consequences too
- Since python 3.10, there are *soft keywords* too (`match`, `case`)
- My simple rule – no keyword contains underscores between words ☺ … yet.

Built-in Functions

A	E	L	R
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	F	M	
ascii()	filter()	map()	S
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	G	N	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	H	object()	
callable()	hasattr()	oct()	T
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		P	V
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	Z
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

Expressions and Statements

Australian National University

- Expression: a combination of values, variables and operators

```
>>> 42
42
>>> n
17
>>> n + 25
42
```

- Statement: code that has an effect or makes a change to state

```
>>> n = 17
>>> print(n)
```

Downey (2015) Think Python, 2nd Ed.

Numeric operators in python

Australian National University

Operator	Function
`+`, `-`, `*`, `/`	Standard arithmetic
`**`	Power (`x ** n` means X^n)
`//`	Floor division (`9 // 2` gives `4`)
`%` (modulus)	Remainder (`9 % 2` gives `1`)

- If you have python already installed, try some of these out (with iPython through a terminal, or with Spyder via the console)
- Some operators can be applied also to values of other (non-numeric) types, but with a different meaning. This is called **operator overloading**.
- We'll see more operators later in the course.

Floor division and modulus

Australian National University

- Floor division is a neat trick in Python. It divides two numbers, discards the remainder and returns an integer value:

```
>>> minutes = 105
>>> hours = minutes // 60
>>> hours
1
```

- To get the remainder, use the modulus '%':

```
>>> remainder = minutes % 60
>>> remainder
45
```

Downey (2015) Think Python, 2nd Ed.

Order of operations - Precedence

Australian National University

- You should know this from high school maths
- The order of precedence of mathematical operators
- PEMDAS: parentheses, exponents, multiplication, division, addition, subtraction
- Matters a whole lot in code – PEMDAS is strictly enforced:

  The result of $6 + 4 / 2^2 - 2 * 10$ is very different to $(6 + 4) / (2^2 - 2) * 10$
- If in doubt, just use parentheses. Some overuse of parentheses is much better than coding a bug that is very hard to track down

Comparison operators

Australian National University

Operator	
<, >, <=, >=	ordering
==	equality
!=	Not equal

- Can compare two values of the same type (for almost any type)
- Comparisons return a *truth* value (type `bool`), which is either `True` or `False`
- *Caution*: Conversion from any type to `bool` happens automatically, but the result may not be what you expect.

```
>>>
>>> if "False":
...     print("Really?")
...
Really?
>>>
```

* *Don't compare* `float`**s with** `==`

String operations

Australian National University

- Funny use of mathematical ‘+’ and ‘*’ operators on strings
- This is common syntactic shorthand in many languages, but the specifics differ from language to language
- In Python:

```
>>> sentence = ‘This’ + ‘is’ + ‘a’ +’sentence’
>>> print(sentence)
Thisisasentence

>>> ‘a’ * 3
aaa

>>> (‘a’ + ‘b’ + ‘c’) * 3
abcabcabc
```

Comments

Australian National University

- Use them! It is good programming practice
- Sensible use of comments throughout your code is a good habit to cultivate
- Makes your code easier to read and be understood by others
- Will help you remember what you did
- Can start as a structure to guide your coding – like writing pseudocode

- If you ever become part of larger, group-programming projects, your commenting style will really begin to matter.

Comments: The # symbol

Australian National University

- All lines in python, and many other languages, the remainder of a line following a '#' will be ignored by the interpreter/compiler

```
# compute the percentage of the hour that has elapsed
percentage = (minute * 100) / 60
```

```
percentage = (minute * 100) / 60     # percentage of an hour
```

Downey (2015) Think Python, 2nd Ed.

Multi-line comments ”””

Australian National University

- Sometimes is it useful to have comments that are a paragraph of text
  - Using # at the beginning of every line can become annoying

- Instead, bound the paragraph with three “ symbols together:

```
”””     Here is a multi-line comment that allows
        a block of text
        so I can waffle on about my code design
        or include good usage notes for my script  """
```

Types of errors

Australian National University

- Every beginning (and experienced) programmer makes errors!
- Learning to remain calm and know what to do when you see one is part of the programmer skillset. Python is quite chatty
- These can be classified into:
- Syntax errors – the language usage you have written is not understood
- Runtime errors – an exception is raised. Your code is legal, but at runtime something unexpected occurred.
- Semantic errors – Your code runs without error, but does the wrong thing.

Exercises

Australian National University

- Complete Exercises 2-1 and 2-2 of Think Python.

Reading

- Chapter 2 of *Think Python* AND/OR
- Chapter 1 of *Automate the Boring Stuff with Python*

Intermission: Installing python with Anaconda

You must try this before labs start next week

Australian
National
University

Installing python

Australian National University

- Every book about beginning python starts with ‘installing python’
- Some computer OS have python installed by default, others don’t. Some will only have python 2 installed.
- We will solve many difficulties if we all install python with Anaconda

Anaconda?

- The cool thing about python is the number of external code libraries you can use.
- A less-cool thing about python is the job of installing the libraries you want AND all of their dependencies. This is where Anaconda helps out (and mostly makes is easier…).
- For now, we will use Anaconda to install a recent release of python and a couple of IDES

Install instructions:

Australian National University

- The long description is here: https://comp.anu.edu.au/courses/comp1730/labs/install/

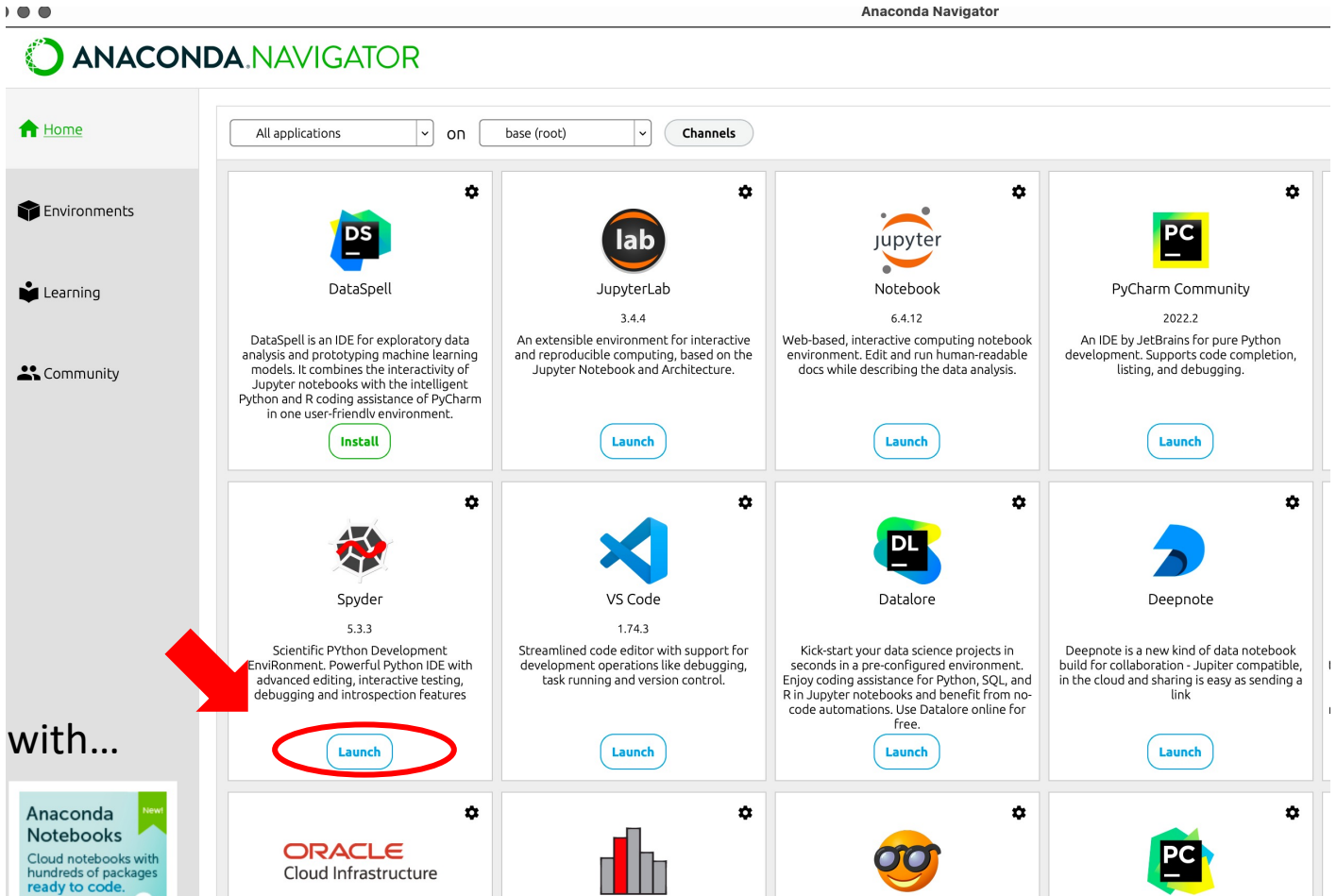
- The short description is - go to: https://www.anaconda.com/products/distribution

- If you get stuck:
  - Consider attending the CSSA Install-Fest – Friday 24th at 5pm onwards.
  - CSIT Building, rooms N109, N112 from 5PM onwards and N113 and N114 after 6PM
  - Or, the Week 2 labs will check your installation – and maybe fix things that went wrong

Anaconda comes with many different IDEs:

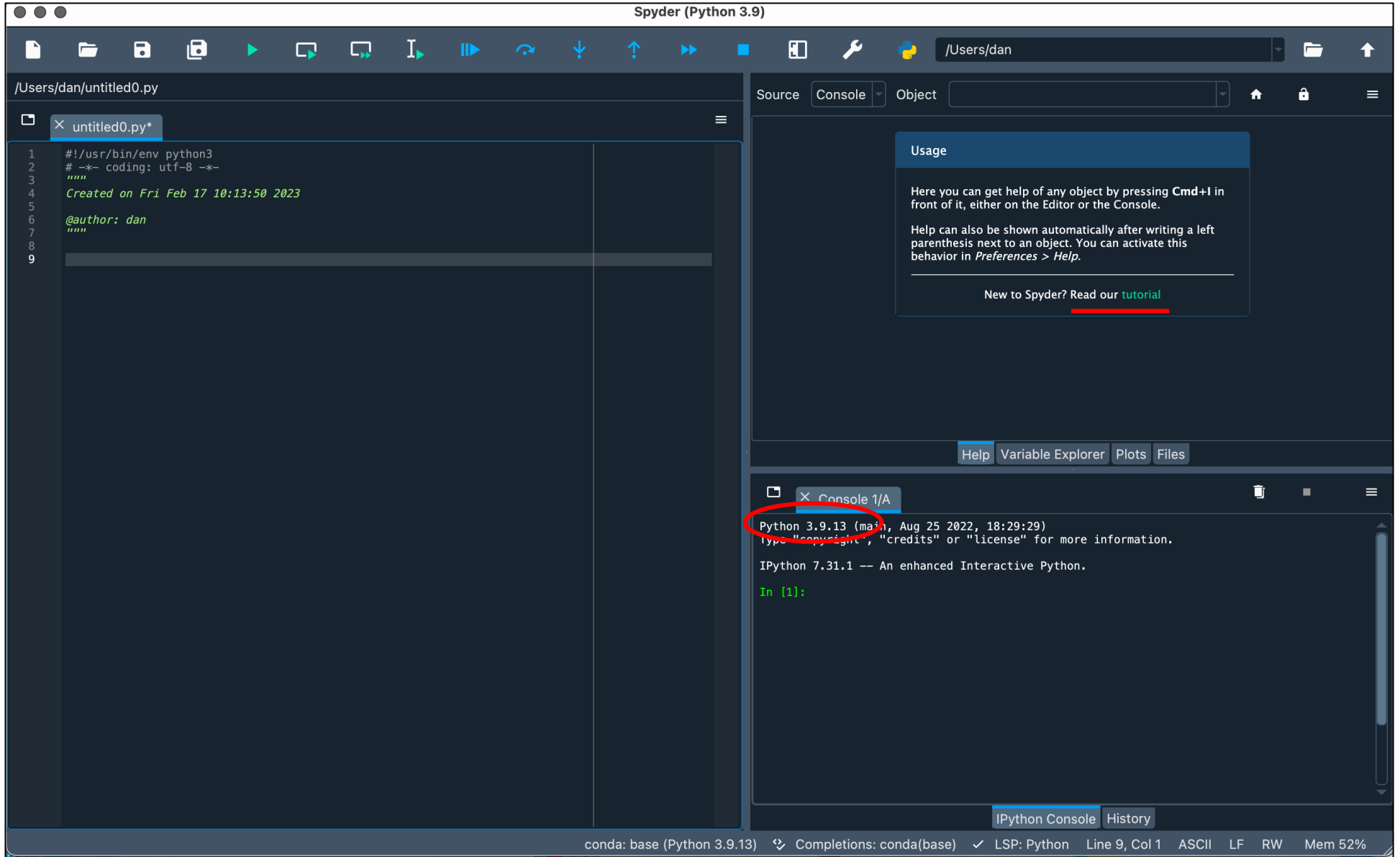
Australian National University

- Spyder 5.3.3 via Anaconda worked for me, but it is a bit flakey.
- VSCode 1.74.3 worked out of the box



This might a SLOW to start with…

Spyder



Australian National University

Check the Python version you get with Anaconda.

I got 3.9 a month back. Obvs the installer used what was already on my laptop?

Should be as new or newer than this.

Intro to Functions (part I)

COMP1730/3730

Reading: Textbook chapter 3 : Alex Downey, *Think Python*, 2nd Edition (2016)

Australian National University

Functions (*Think Python*, Ch. 3)

Australian National University

- Functions are like mini-programs you can call from your code that do useful, predefined tasks (that you would otherwise you might need to do for yourself)
- We have already seen two:
  - str(0.1) converts a number (integer or float) to a string
  - print('this') takes a string input and prints it to the terminal
- In python, functions can be:
  - Built-in
  - Imported from modules
  - User-defined

Built-in Python functions

Australian National University

- There LOTS of these. A very necessary part of the language
- Go to the source documentation for the Python language (at python.org) and look through what is available:

https://docs.python.org/3/library/functions.html

- Each function is described with details of what it does, what input it takes and what output it produces

Built–in Functions

A	E	L	R
abs()	enumerate()	len()	range()
aiter()	eval()	list()	repr()
all()	exec()	locals()	reversed()
any()			round()
anext()	F	M	
ascii()	filter()	map()	S
	float()	max()	set()
B	format()	memoryview()	setattr()
bin()	frozenset()	min()	slice()
bool()			sorted()
breakpoint()	G	N	staticmethod()
bytearray()	getattr()	next()	str()
bytes()	globals()		sum()
		O	super()
C	H	object()	
callable()	hasattr()	oct()	T
chr()	hash()	open()	tuple()
classmethod()	help()	ord()	type()
compile()	hex()		
complex()		P	V
	I	pow()	vars()
D	id()	print()	
delattr()	input()	property()	Z
dict()	int()		zip()
dir()	isinstance()		
divmod()	issubclass()		_
	iter()		__import__()

Example Built-in functions:

`print(), len(), round(), input()`

Australian National University

- As a useful exercise, go to the python.org documentation for built-in functions and look up these up

```
>>>
>>> print('Some text here')
Some text here
>>> len('Some text here')
14
>>> round(1.1)
1
>>> round(1.9)
2
>>> input_string = input()
here is something I typed
>>> print(input_string)
here is something I typed
>>>
```

Imported Functions

Australian National University

- These work a lot like built-in functions, but need to be imported first and called with reference to the module they come from.
- Sometimes modules are referred to as packages or libraries
- A full list of Python modules is available at python.org:

https://docs.python.org/3/py-modindex.html

- Have a look at what is available – lots of useful stuff, eg:

math	json	pickle
random	statistics	getopt
zlib	pprint	csv

Python Module Index

_ | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | z

_	
__future__	*Future statement definitions*
__main__	*The environment where top-level code is run. Covers command-line interfaces, import-time behavior, and ``__name__ == '__main__'``.*
_thread	*Low-level threading API.*
a	
abc	*Abstract base classes according to :pep:`3119`.*
aifc	**Deprecated:** *Read and write audio files in AIFF or AIFC format.*
argparse	*Command-line option and argument parsing library.*
array	*Space efficient arrays of uniformly typed numeric values.*
ast	*Abstract Syntax Tree classes and manipulation.*
asynchat	**Deprecated:** *Support for asynchronous command/response protocols.*
asyncio	*Asynchronous I/O.*
asyncore	**Deprecated:** *A base class for developing asynchronous socket handling services.*
atexit	*Register and execute cleanup functions.*
audioop	**Deprecated:** *Manipulate raw audio data.*
b	
base64	*RFC 4648: Base16, Base32, Base64 Data Encodings; Base85 and Ascii85*
bdb	*Debugger framework.*
binascii	*Tools for converting between binary and various ASCII-encoded binary representations.*

Module functions example: math

Australian National University

- The functions available in modules would be rather annoying to have to write from scratch each time: math.log10(), math.sin(), etc
- To use a particular module, one must first import it:

```
>>> import math
```

- Then to use the functions, one must use dot notation:

```
>>> ratio = signal_power / noise_power
>>> decibels = 10 * math.log10(ratio)

>>> radians = 0.7
>>> height = math.sin(radians)
```

Downey (2015) Think Python, 2nd Ed.

Exercises

Australian National University

- Complete Exercises 3-1, 3-2 and 3-3 of *Think Python*.

Reading

- Chapter 3 of *Think Python* AND/OR
- Chapter 3 of *Automate the Boring Stuff with Python*