Useful Python Features

Unusual Operations:

// = integer division

/ = floating point division

\*\* = raise to the power of

^ = XOR bitwise operator

== tests equivalence of 2 objects

=! tests inequivalence of 2 objects

Eg print(3==3) returns True

Print(3!=3) returns False

Functions:

Type – Returns data type of input

Print – prints input

Str – converts input into string

Len – Gives length of a string

Slicing – Gives portion of a string eg string1[0:4] gives the characters indexed from 0 to 4 of the string

Digits – Gives a string containing all possible digits

Uppercase – Gives a string containing all possible uppercase letters

Letters – Gives a string containing all possible lowercase and uppercase letters

Index – Gives index of substring within larger string

Eg sentence = 'Python programming is fun.'

result = sentence.index('is fun')

print("Substring 'is fun':", result)

Substring 'is fun': 19 (index number of I in ‘is fun’)

Data types:

String – eg ‘Hello’

Integer – eg 5

Float – eg 5.0

Bool – True or False

Lists – Arrangements of values identified with an index eg [-2, ‘cat’, False]

Lists can contain data of different types

To extract a specific item of a list, use the index List = [1, 2] List[0] = 1

NB Lists and tuples are indexed from 0

Tuples – Lists that cannot be changed, use round brackets unlike the square brackets of lists

Mutable vs Immutable data:

Mutable objects are objects whose data can be changed, once defined eg lists

Immutable objects are objects whose data cannot be changed once define eg strings, tuples

Useful operations/logicals with conditions statements:

==

>

<

!=

=>

<=

and

or

not

Libraries:

math – used to call mathematical constants eg print(math.pi) = 3.14159…

pyplot – used to plot data

eg import matplotlib.pyplot as plt

define 2 lists

plt.plot(list1, list2)

plt.show() This will plot a graph of list 2 against list 1, alternatively, define the

lists within the plt.plot function

To specific sub-elements of a list within a list, write the index number of the sub-list within the overall list, then the index number of the element of that sub-list

Eg ﻿list1 = [1, 2, 3, 4, 5, 6, 7]

list2 = [16, 2, 43, 22., .11116, 2./3., 72.]

list3 = ['a', 'b', '63']

list4 = ['abcde', '54']

list5 = []

list5.append(list1)

list5.append(list2)

list5.append(list3)

list5.append(list4)

for i in range(len(list5)):

print(list5[i][0], list5[i][-1])

returns

﻿1 7

16 72.0

a 63

abcde 54

Operator overloading = When operations have different effects depending on the data type of the argument passed to them

Eg 2\*2 = 4

‘Hello’ \* 2 = ‘HelloHello’

Docstring = Text highlighted at top of program holding text comments about the program

Docstring is housed between 2 “””

NB Python can only perform operations on data of the same type

Pygame:

pygame.init() initialises pygame

pygame.display.set\_mode((a,b)) creates our game surface at a size of a px by b px, note we pass the tuple (a,b) as an argument to the function, if we pass a and b separately, an error will occur