Python tutorial notes:

Rock paper scissor:

Key parts:

The random module is ues to randomly select the computers choice between rock paper or scissors. Furthermore outside of rock paper scissors it can be used to output a random value, percentage etc.

Input allows the player to pick a choice, in this context it was user to pick between rock paper or scissors and then store that value for the player until the computer was to have their choice chosen.

Dictionaries are used to store a players choice and computers choice in this context it was used like this :

Choices = {“player”: player_choice, “computer”: computer_choice}

Conditional statements are used to compare choices and determine a winner, these are:

.if

.elif

.else

Finally with the use of an fstring, it will then output the winner of one game of rock paper scissors:

print(f"You chose {player} and computer chose {computer}")

Basics:

Variables and data types:

str – strings

int – integers

float – floating-point numbers

bool – True/False

These are some of the ways to identify the different data types in python

Operators:

Arithmetic: +, -, *, /, %, **, //

Comparison: ==, !=, >, <, >=, <=

Boolean: and, or, not

Bitwise: &, |, ^, ~, <<, >>

Ternary: shorthand for conditional expressions:

These are all the different types of operators, the code with in my python repository shows how I have used them with in code.

Strings:

Any word is string such as names

Some of the string methods we user are :

#isalpha() to check if a string contains only characters and is not empty

isalnum() to check if a string contains characters or digits and is not empty

isdecimal() to check if a string contains digits and is not empty

lower() to get a lowercase version of a string

islower() to check if a string is lowercase

upper() to get an uppercase version of a string

isupper() to check if a string is uppercase

title() to get a capitalized version of a string

startsswith() to check if the string starts with a specific substring

end swith() to check if the string ends with a specific substring

replace() to replace a part of a string

split() to split a string on a specific character separator

strip() to trim the whitespace from a string

join() to append new letters to a string

find() to find the position of a substring

Slicing is when you will only output a certain amount of characters with in a word you can do this by writing:

Name = Taran

Name[0 : 2]

Output:

Tar

Concatenation is when you essentially add two different words and put them together into a sentence:

Name = Taran

Print (“Hello” + Name + “!”)

Output:

Hello Taran!

Lists:

Lists are ordered or unordered collection of similar data

You can also perform slicing on list as well:

```
dogs = ["Roger", 1, "Syd"]

print(dogs[1:3])
```

output:

[1, ‘Syd’]

Tuples:

Tuples are similar to lists but they are immutable, which basically means once you have made one, you are unable to and cannot change their content.

This means you are unable to concatenate tupes

Dictionaries:

These are a collection of key value pairs u rapidly look up a vlue using a unique key.

Useful methods: .keys(), .values(), .pop(), .popitem()

Sets:

These are unordered collections of unique elements

Functions:

Nested functions – functions inside functions

Closures – inner function remembers outer variables

Lambda – anonymous functions: multiply = lambda a, b: a * b

Control flow:

If-else: conditional branching

Loops:

while loop – repeat until a condition is False

for loop – iterate over collections

break – exit loop early

continue – skip current iteration

Classes and objects

Class defines attributes and methods

Inheritance:

A class can inherit methods from other classes

Polymorphism

Different clases can implement the same method differently

Operator overloading

Customize operators for objects

Add() respond to the + operator
sub() respond to the - operator
mul() respond to the * operator
truediv() respond to the / operator
floordiv() respond to the // operator
mod() respond to the % operator
pow() respond to the ** operator
rshift() respond to the >> operator
lshift() respond to the << operator
and() respond to the & operator
or() respond to the | operator
xor() respond to the ^ operator

File handling:

Reading/ writing files

Exceptions:

Catch errors to prevent crashes

Decorators:

Functions that wrap other functions

Recursion:

A type of loop where functions call them self until they meet a base condition

Docstrings:

Document functions and classes

Map, filter and reduce:

Map: applies a function to every item in a list.

Filter: keeps only the items that meet a condition.

Reduce: combines all items into a single value using a function.

Modules and pip:

Import built in or third party libaries