Python quick notes

Str > “” “can carry numbers” “ can carry letters and symbols”
Int > 354749 # numbers only no decimals
Float > 34567.789 #can have decimals
Bool > True False
Data type conversion > Int() str() float() bool()
Indexing [1]
Slicing [1:2]
List [“this”, “is”, “a”, “list”] > .sort(), .extend(), .append(), .insert(index, item), .remove(), .pop(), max(lst), min(lst), .reverse() or [::-1]
= setting variable
+= add this item
-= subtracting this item
/= dividing this item (decimal)
%= finding leftover of this item
//= dividing item (drops decimal)
**= power
Check logic a, b = True, False
a and b > False
a or false > True
not a > False
Arithmetic
x, y = 10, 3
x + y = 13
x - y = 7
Etc.
Identify
x = [1,2,3]
y = x
z = [1,2,3]
x is y > True
x is z > False #has same numbers but they aren’t one another like y is x use == instead for True
Containment
Nums = [1,2,3,4]
3 in Nums > True
5 in Nums > False
Comparison
== #equal to
>= #greater than or equal
<= #less than or equal to
!= #not equal to
If # if this is whatever
Elif #same as if statement but comes after

Else #other situation if , “if statement” does not work
Nested two sets of if statements if this : and separate lines if this: print this
Compound using comparisons above and and statement
While loops # while this condition is true, false, whatever
For loops # for this / or in / or range etc
Break # stops loops
Continue # continue code after interaction for i in range(5) i = 2 continue prints [1,3,4,5] #takes out two
While loops #can have if statements and conditions (compounds too)
For loops #can have if statements and conditions (compounds too)
Pass #placeholder does nothing for i in range(5) i = 2 pass prints [1,3,4,5]

Files >
Opening files

```
File = open(“example.txt” , “w”) #opens the file #”w” write
File.write(“Hello, World!”) #writes in file
File.close()  #closes the file
```

Reading files

```
File = open(“example.txt” , “r”) #opens the file #”r” read
content = file.read() #reading file
print(content) #shows what was read
file.close() #closes file
```

Writing to a file

```
File = open(“example.txt”, “a”)
File.write(“\nAppended Text.”)
File.close
```

Checking existence of a file

```
os.path.exists(“example.txt”):
if os.path.exist(“example.txt”):
	print(“file exists”)
else:
	print(“File nonexistent”)
```

Deleting a file

```
Import os
If os.path.exist(“exmaple.txt”):
	os.remove(“example.txt”)
	print(“File deleted”)
else:
	print(“file does not exist”)
```

Using with statement > handles opening and closing file so you don't have to worry about closing manually

Writing
```
with open(“example.txt”, “w”) as file:
        file.write(“Hello from ‘with’ !”)
```
Reading
```
With open(“example.txt”, “r”) as file:
        Content = file.read()
        print(Content)
```
“r” = read #file must exist
 “w” = write #deletes previous stuff and overwrites
 “a” = appended #adds to files content

string.format() - {add words/variable here}
 f’string - {add words/variable here}

```
#this is a comment
        This is indented
“””

This is white space

“””
```
Save code in example.py
To run file run pydoc example in the terminal to see the documentation generated form the docstring

Signature (way to call function)
Default values (b=5) b is 5
Def - defining a function
Pass - placeholder

Syntax errors occur when code does not follow python rules (to correct fix structure)
Logic errors - runs but wrong result (to correct , revisit reasoning and correcting steps)
runtime error - happen when code is running because (typically) invalid operations (to correct add checks like to avoid division by 0)

Try block - to try a excerpt of code
Except block - exception if try does not work
Else - if try works
Finally block - runs regardless
Raise - intends to raise except block

Writing a function and test with unittest

```
def add_numbers(a, b):
        Return a + b


import unittest
Class TestAddNumbers(unittest.TestCare):

        def test_add_numbers(self):
                self.assertEqual(add_numbers(3, 5), 8) #check equality Checks if a == b
                self.assertTrue(add_numbers(1, 1),0) #check if true Passes if expr is True
                self.assertIsInstance(add_numbers(2, 3), int) #check type  if a is b (identity, not just value)
                self.assertIs(add_numbers(0, 0), 0) #check identity if obj is an instance of cls
                self.assertIn(add_numbers(1, 2), [3, 4, 5]) #check membership if item is in collection
If __name__ == "__main__":
        unittest.main()
```

Output would be

```
Ran 1 Test in 0.001s
OK
```

import os - provides functions with operating systems
import os.path - helps handle file paths
import sys - for system parameters and functions line command lines

```
file_name = "example.txt"
if os.path.exists(file_name):
        #check for existence

with open(file_name, "w") as file:
        file.write("Hello, world!")
        #create and write to the file

with open(file_name, "r") as file:
        content = file.read()
        print("File content:", content)
        #read file
```

Command line arguments

```
import sys
if len(sys.argv) > 1:
        print(“arguments passed;” , sys.argv[1:])
else:
        print(“No arguments provided.”)
```

```
import math
print(math.fabs(-5.5)) #5.5

print(math.ceil(4.2) #5
print(math.floor(4.8) #lowest int
print(math.trunc(4.8)) #4  #drops decimal

#modulo %
print(math.fmod(10, 3)) #1.0 (remainder)

print(math.frexp(16)) #(0.5, 5) is the mantissa*exponent of the number
print(math.sqrt(25)) #5.0 (sq decimal)
print(math.issqrt(25)) #5 (sq int)
print(math.pwo(2,3)) #2^3 pr 8.0

print(math.pi) #pi

print(math.isnan(math.nan)) #check if value is NaN
```

Datetime module

```
from datetime import datetime
now = datetime.now() #date time
print(now)

formatted = now.strftime("%Y-%m-%d %H:%M:%S")
print(formatted) #year, month, day, hour, min, secs

print(now.weekday())  #day of the week
```

Random module

```
import random

print(random.randint(1,10)) # random int 1- 10
print(random.randrange(1,10)) # random int excludes 10
```