



Higher Python Quick Guide

File Handling – Parallel Arrays

Writing a String to a line in a File

```
with open("newfile.txt","w") as writefile:
    writefile.write("Python is great fun"+ "\n")
```

Reading all of a plain text file (in one operation)

```
with open("SampleFile.txt") as readfile:
    filecontents = readfile.read()
```

Reading a CSV file into parallel arrays

```
counter = 0
names = [None] * 50
marks = [None] * 50
with open("marks.csv") as readfile:
    line = readfile.readline().rstrip('\n')
    while line:
        items = line.split(",")
        names[counter] = items[0]
        marks[counter] = items[1]
        line = readfile.readline().rstrip('\n')
        counter += 1
```

File Contents

```
Joe, 42
Anna, 90
```

Creating a CSV file

```
for counter in range(5):
    name = input("Enter name: ")
    age = input("Enter age: ")

    with open("names.txt","w") as writefile:
        writefile.write(name + "," + str(age) + "\n")
```

Subroutines

Defining a subroutine

```
def addnumbers():
```

Returning a value from a subroutine

```
def getInput():
    number1 = int(input("Enter Number"))
    number2 = int(input("Enter Number"))
    return number1,number2
```

Defining formal parameters for a subroutine

```
def addnumbers(val1,val2):
    answer = val + val2
    return answer
```

Calling a subroutine (and passing actual parameters)

```
displaydetails(answer)
```

Assigning return values from a subroutine

```
number1,number2 = getInput()
```

Predefined Functions

Modulus Function – finds the remainder

```
remainder = 5 % 2
print(remainder)
>>> 1
```

Converting a Character to ASCII

```
print(ord("a"))
>>>97
```

Converting ASCII code to Character

```
print(chr(65))
>>>A
```

Converting a REAL number to INTEGER

```
original = 24.54
newnumber = int(original)
print(newnumber)
>>>24
```

Standard Algorithms

Linear Search

```
found = False
choice = input("Please enter the name to search for: ")
for counter in range(len(names)):
    if names[counter] == choice:
        found = True
        position = counter
if found == False:
    print("No item found in list")
else:
    print("Item details: ",names[position])
```

Finding Minimum

```
minimum = scores[0]
for counter in range(1,len(marks)):
    if marks[counter] < minimum:
        minimum = marks[counter]
print("Lowest score is", minimum)
```

< to find the min

> to find the max

Finding Minimum with position

```
minpos = 0
for counter in range(1,len(marks)):
    if marks[counter] < marks[minpos]:
        minpos = counter
print("Lowest score:", marks[minpos],"by",marks[minpos])
```

Count Occurences

```
occurrences = 0
for counter in range(len(marks)):
    if marks[counter] >=50:
        occurrences += 1
print("There were",occurrences,"passes")
```



Higher Python Quick Guide

Substrings

0	1	2	3	4	5	6	7	8	9	10	11	12
I		a	m		a		s	t	r	i	n	g

-13	-12	-11	-10	-9	-8	-7	-6	-5	-4	-3	-2	-1
I		a	m		a		s	t	r	i	n	g

Negative Indexes – if right of string is needed.

Extracting a substring example 1

```
print(original[2:4])
am
>>>
```

Extracting a substring example 2

```
print(original[7:len(original)])
string
>>>
```

Selecting characters from end of string

```
print(original[-4])
r
>>>
```

Checking a single character

```
for x in range(len(original)):
    if original[x] == " ":
        count += 1
print(count)
>>>
```

Can also be

```
print(original[7:])
string
>>>
```

Extracting every second character

```
print(original[::2])
Ia tig
>>>
```

Reading a CSV file into an array of records

```
counter = 0
pupils = [pupil() for x in range(40)]
with open("marks.csv") as readfile:
    line = readfile.readline().rstrip('\n')
    while line:
        items = line.split(",")
        pupils[counter].name = items[0]
        pupils[counter].mark = items[1]
        line = readfile.readline().rstrip('\n')
        counter += 1
```

Creating a CSV file using an array of records

```
with open("names.txt", "w") as writefile:
    for x in range(len(pupils)):
        writefile.write(pupils[x].name + "," + str(pupils[x].mark) + "\n")
```

Record Structure Used

All record examples will use this record structure

@dataclass

```
class pupil:
    name: str = ""
    mark: int = ""
```

Array of Records

Declaring a Record

```
from dataclasses import dataclass
```

@dataclass

```
class pupil:
    height: float = 0.0
    weight: float = 0.0
    bmi: float = 0.0
```

Creating an Array of Records

```
myBMI = [BMI() for x in range(40)]
```

Populating an Array of Records

```
for x in range(len(myBMI)):
    myBMI[x].height = float(input("Enter Height: "))
    myBMI[x].weight = float(input("Enter Weight: "))
```

Standard Algorithms using Array of Records

Linear Search

```
found = False
choice = input("Please enter the name to search for: ")
for counter in range(len(pupils)):
    if pupils[x].name == choice:
        found = True
        position = x
if found == False:
    print("No item found in list")
else:
    print("Item details: ", pupils[position].name)
```

Finding Minimum with position

```
minpos = 0
for x in range(1, len(pupils)):
    if pupils[x].mark < pupils[minpos].mark:
        minpos = x
print("Lowest mark:", pupils[minpos].mark, "by", pupils[minpos].name)
```

Count Occurrences

```
occurrences = 0
for x in range(len(pupils)):
    if pupils[x].mark >= 50:
        occurrences += 1
print("There were", occurrences, "passes")
```

Finding Minimum

```
minimum = scores[0]
for x in range(1, len(pupils)):
    if pupils[x].mark < minimum:
        minimum = pupils[x].score
print("Lowest mark is", minimum)
```