

# 1 Introduction

```
1 my_number = 10
2 print(type(my_number))
3 my_number = 10.0
4 print(type(my_number))
5 my_number = "10"
6 print(type(my_number))
7
8 my_number = "ten" + "hello"
9 print(my_number)
```

Listing 1: Python example

```
Python 3.7.2 (v3.7.2:9a3ffc0492, Dec 24 2018, 02:44:43)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: /Users/Paul/Desktop/code_activities/first_class.py ======
<class 'int'>
<class 'float'>
<class 'str'>
tenhello
>>> |
```

Output

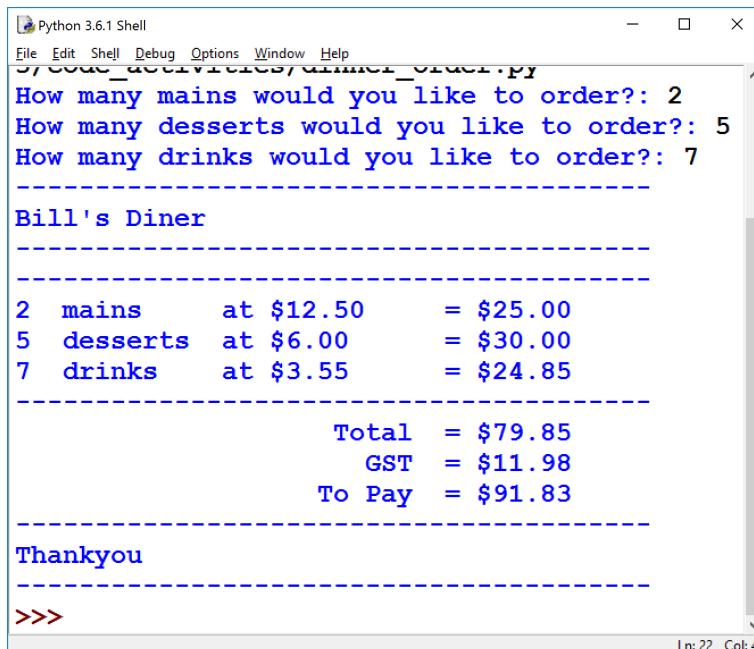
## 2 Basic Program

### 2.1 Dinner Order

You need to create a program that allows a waiter to enter in the number of main meals ( at \$12.50 each ) , the number of desserts ( at \$6.00 each ) and the number of drinks ( at \$3.55 ) ordered at a table.

The program then prints out a summary of the order, a total and then adds on GST (and gives the new total ).

(Note: the program should be easy to alter so the price values should be declared as constant variables)



A screenshot of the Python 3.6.1 Shell window. The title bar says "Python 3.6.1 Shell". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The code editor pane shows a file named "dinner\_order.py" with the following content:

```
How many mains would you like to order?: 2
How many desserts would you like to order?: 5
How many drinks would you like to order?: 7
-----
Bill's Diner
-----
2 mains      at $12.50      = $25.00
5 desserts    at $6.00       = $30.00
7 drinks      at $3.55       = $24.85
-----
          Total   = $79.85
          GST    = $11.98
          To Pay = $91.83
-----
Thankyou
-----
>>>
```

The status bar at the bottom right indicates "Ln: 22 Col: 4".

An example of Dinner Order program run

#### 2.1.1 Planning

For the writing and planning:

- **Decomposition** Please write down your task decomposition.

(This is breaking the task down into smaller pieces that will be combined to make the finished program)

See examples below.

- **Version Log**

Your version log should go here. Annotated screenshots are a good idea at this point

- **Component Trialling**

Show that you have trialled each component here.

You should also include notes that justify the major decisions you made.

## • Assembled Outcome Testing

Please show testing for your assembled outcome below.

This should include a test plan followed by screenshot proof

## • Usability Testing

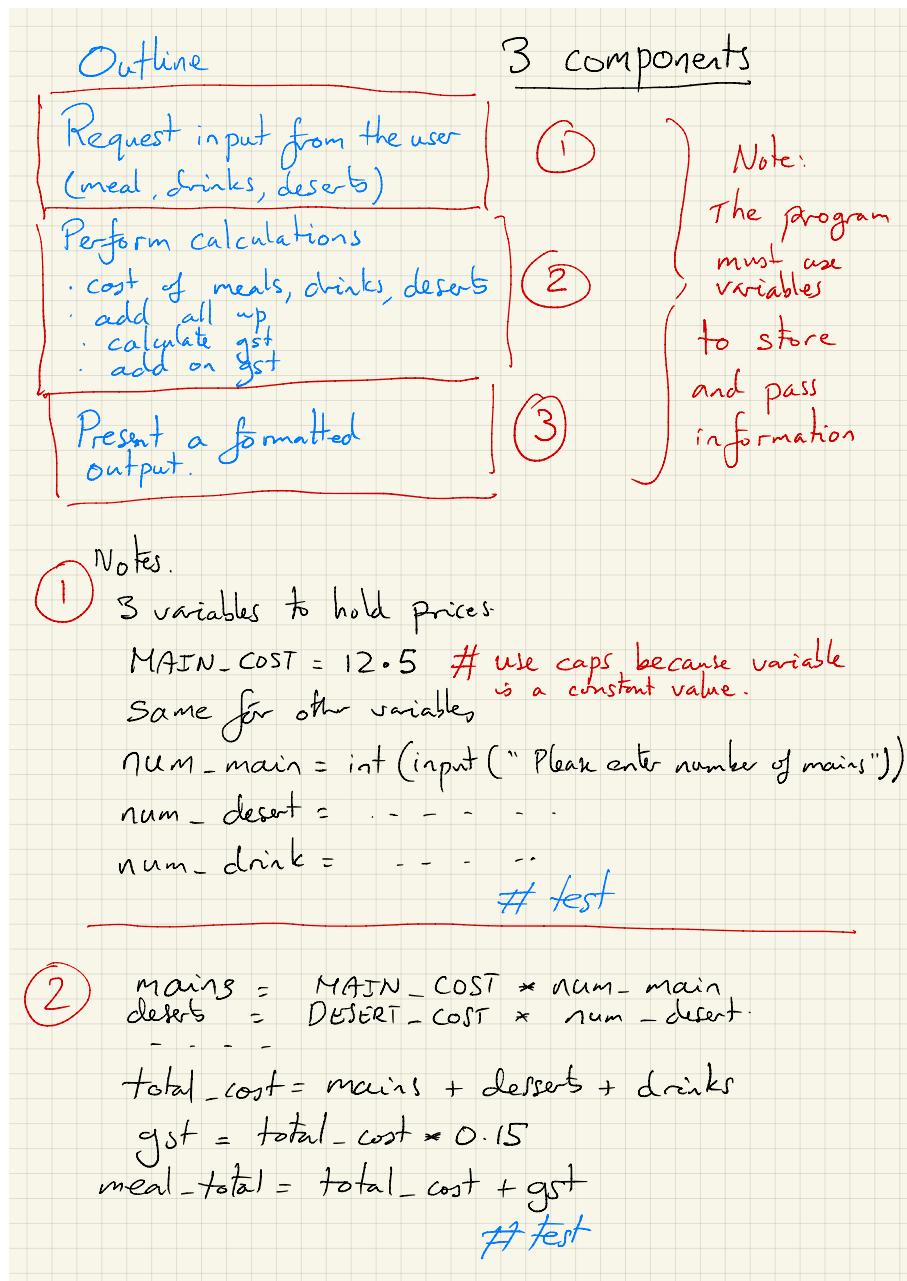
Try the program out on someone else and make notes or video etc. Write a list of things improvements which need to be made based on your usability testing. Then write down what you changed.

## • General Evaluation:

How good is my program? What can I do to improve it?

## Examples of Component Planning

**Basic Idea of Testing** When testing we consider



(3)

3 mains at \$12.50 = \$37.50  
 2 drinks at \$ -- = --  
 5 deserts at \$ -- = --

line-one = "{ } mains at \${ } = \${ }". Sub-total = --.  
 :3 format(num\_main, MAIN-COST, to pay = --.)  
 :6.2f .2f

- Expected inputs (these are inputs we would normally expect the user to input).

Does the program give the right outputs with the given expected inputs.

- Boundary inputs (are there maximum or minimum value that the user can enter)

What happens at these boundaries (and if we go over them)

- Unexpected Inputs (these are character entries that we are not expecting)

These could be just pressing enter or having spaces , using letters instead of numbers or characters like \* , &, etc.

These could occur if the user makes a mistake or misunderstands what is expected.

Testing should have some kind of plan and documentation of the results.

Testing dinner order. (Done once the program is made, or a version of it)

Expected Input      Boundary Input      Unexpected Input

How many mains ? 4	How many mains ? 30	How many mains ? Three
How many deserts ? 3	How many deserts ? 0	How many deserts ? *?
How many drinks ? 2	How many drinks ? -1	How many drinks ? _____ ↑ space.

Expected Output ?      Expected Output ?      Expected Output ?

Actual Output ?      Actual Output ?      Actual Output ?

is this okay ?

is this okay ?

is this okay ?

## Result from expected

```
>>> 
= RESTART: /Users/Paul/Desktop/code_activities/trinket/dinner_order_basic.py
How many mains would you like to order?: 4
How many desserts would you like to order?: 3
How many drinks would you like to order?: 2
-----
Bill's Diner
-----
4 mains at $12.50 = $50.00
3 desserts at $6.00 = $18.00
2 drinks at $3.55 = $7.10
-----
Total = $75.1
GST = $11.26
To Pay = $86.36
-----
Thankyou
-----
>>> |
```

## Result from boundary

```
>>> 
= RESTART: /Users/Paul/Desktop/code_activities/trinket/dinner_order_basic.py
How many mains would you like to order?: 30
How many desserts would you like to order?: 0
How many drinks would you like to order?: -1
-----
Bill's Diner
-----
30 mains at $12.50 = $375.00
0 desserts at $6.00 = $0.00
-1 drinks at $3.55 = $-3.55
-----
Total = $371.45
GST = $55.72
To Pay = $427.17
-----
Thankyou
-----
>>> |
```

## Result from unexpected

```
type 'help', 'copyright', 'credits' or 'license()' for more information.
>>>
= RESTART: /Users/Paul/Desktop/code_activities/trinket/dinner_order_basic.py =
How many mains would you like to order?: Three
Traceback (most recent call last):
  File "/Users/Paul/Desktop/code_activities/trinket/dinner_order_basic.py", line 1
  3, in <module>
    mains = int(input("How many mains would you like to order?: "))
ValueError: invalid literal for int() with base 10: 'Three'
>>>
```

## Evaluation

- What works
  - Works properly on expected inputs
  - Presentation of receipt clearly laid out on console.
- What could be improved
  - Should have an upper limit on how many meals can be entered
  - Should not allow entries below zero
  - Unexpected inputs cause a program crash
  - At the moment the program only runs once (so need a way to enter a new order)

Components: (parts of the program we need to  
 be able to make).  
 (The components will then be assembled to make  
 the finished program).

- ① Getting the user input
- ② Calculating the value of the meal
- ③ Printing out the receipt.

① *variable*

```
num_mains = int(input("How many mains would you like?"))
num_deserts = int(input("How many deserts would you like?"))
num_drinks = int(input("How many drinks would you like?"))

print(num_mains)
print(num_deserts)
print(num_drinks)
```

*polite, clear message to user.*

*print out to test that it is working*

Save with logical name: dinner\_input\_component

② MAIN = 12.5  
 DESERT = 6.0  
 DRINK = 3.55

num\_mains = 3  
 num\_deserts = 2  
 num\_drinks = 4

*Declare the prices as variables.  
 UPPERCASE means will not change (constant).*

*just give the variable values for trialling purposes.*

```
main_cost = num_mains * 12.5
desert_cost = num_deserts * 6.0
drinks_cost = num_drinks * 3.55
```

```
total_cost = main_cost + desert_cost + drinks_cost
print(total_cost) ← test.
```

```
gst = total_cost * 0.15
total_inc_gst = total_cost + gst
```

*print as well*

Save with a logical name: dinner\_cost\_component.

just copy from ②

(3)

```

MAIN = 12.5
DESSERT = 6.0
DRINK = 3.55

num_mains = 3
num_deserts = 2
num_drinks = 4

main_cost = num_mains * 12.5
desert_cost = num_deserts * 6.0
drinks_cost = num_drinks * 3.55

total_cost = main_cost + desert_cost + drinks_cost

gst = total_cost * 0.15
total_inc_gst = total_cost + gst

#print receipt
print ("-" * 40)
print ("Bill's Diner")
print ("---")

print ("{:3}{:10} at ${:10.2f} is ${:10.2f} ".format(num_mains,
                                                    "mains", MAIN, main_cost))
    ↗ 3 spaces           ↗ 10 spaces.   ↗ 10 spaces, left align, 2 decimal places.
    ↗ 2 decimal places. ↗ variables go in order, into brackets.

Repeat for deserts and drinks.

#print last part.
print ("{:>27} = ${:10.2f} ".format("Total", total_cost))

Repeat for gst
and total cost including gst.

```

```

1 #constants
2 MAIN_PRICE = 12.5
3 DESSERT_PRICE = 6.0
4 DRINKS_PRICE = 3.55
5
6 #totalling variables (initially declared as 0)
7 mains_total = 0.0
8 desserts_total = 0.0
9 drinks_total = 0.0
10 ext_gst = 0.0
11
12 # input variables
13 mains = int(input("How many mains would you like to order?: "))
14 desserts = int(input("How many desserts would you like to order?: "))
15 drinks = int(input("How many drinks would you like to order?: "))
16
17 #calulate costs
18 mains_total = mains * MAIN_PRICE

```

```

19 desserts_total = desserts * DESSERT_PRICE
20 drinks_total = drinks * DRINKS_PRICE
21
22 ext_gst = mains_total + desserts_total + drinks_total
23 my_gst = ext_gst*0.15
24 my_to_pay = ext_gst + my_gst
25
26 print("-"*40)
27 print("Bill's Diner")
28 print("-"*40)
29 print("{:<3}{:10}at ${:<10.2f}= ${:.2f}".format(mains,"mains", MAIN_PRICE ,
   mains_total))
30 print("{:<3}{:10}at ${:<10.2f}= ${:.2f}".format(desserts,"desserts",
   DESSERT_PRICE, desserts_total))
31 print("{:<3}{:10}at ${:<10.2f}= ${:.2f}".format(drinks,"drinks", DRINKS_PRICE ,
   drinks_total))
32 print("-"*40)
33 print(":>27}={:}{}".format("Total    ", ext_gst))
34 print(":>27}={:,.2f}{}".format("GST    ", my_gst))
35 print(":>27}={:,.2f}{}".format("To Pay    ", my_to_pay))
36 print("-"*40)
37 print("Thankyou")
38 print("-"*40)

```

Listing 2: Dinner Order Program

```

1 # get number of punnets boxes buckets
2 my_punnets = float(input("How many punnets?: "))
3 my_boxes = float(input("How many boxes?: "))
4 my_buckets = float(input("How many buckets?: "))
5
6 #testing
7 #my_punnets = 2.5
8 #my_boxes = 1.75
9 #my_buckets = 3.5
10
11 punnet_weight = my_punnets*0.25
12 box_weight = my_boxes*1.5
13 bucket_weight = my_buckets*7
14
15 total_weight = punnet_weight + box_weight + bucket_weight
16
17 total_price = total_weight * 5
18 print("."*30)
19 print("{:20}{:>6.2f} kg".format("Punnet weight" , punnet_weight))
20 print("{:20}{:>6.2f} kg".format("Box weight" , box_weight))
21 print("{:20}{:>6.2f} kg".format("Bucket weight", bucket_weight))
22 print("."*30)
23 print("{:20}{:>6.2f} kg".format("Total weight", total_weight))
24 print("."*30)
25 print("."*30)
26 print("{:20} ${:>6.2f}".format("Price to charge", total_price))
27 print("."*30)

```

Listing 3: Python example

### 3 Conditions

The solutions to many programming problems require an action to occur only if a particular condition is true.

A condition can only result in true or false.

(These values are called Booleans. True and False are reserved words in the Python language. True and False are values / objects of type bool)

A condition often involves an expression which compares one value with another.

Comparisons between numerical values are made using the same comparison operators that are used in mathematics:

>	greater than
$\geq$	greater than or equal to
<	less than
$\leq$	less than or equal to

and the equality / inequality operators:

$= =$	equal to ( the proper equals operator)
$! =$	not equal to

#### Boundary cases

Human languages are imprecise.

Does "every child over 5" mean "aged 5 and over" or "aged 6 and over"? The difference between  $\geq$  and  $>$  is very important.

If you mean "every child aged 5 and over" you can express this as age  $> 4$  or age  $\geq 5$ .

Any lack of clarity needs to be removed at the design phase. Always clarify before coding. The boundary cases are where errors in programming often occur.

Boundary cases are the values at and just outside the specified limits.

In the case "aged 5 and over" the specified limit is 5 and the age values of 4 and 5 are the boundary cases.

Only one condition can execute

```

my_variable = 89

# example 1
if my_variable < 90:
    print("{} is less than 90".format(my_variable))

#example 2
if my_variable < 89:
    print("{} is less than 89 ".format(my_variable))

#example 3
if my_variable < 89:
    print("{} is less than 89".format(my_variable))
else:
    print("{} is not less than 89".format(my_variable))

#example 4
if my_variable < 89:
    print("{} is less than 89".format(my_variable))
elif my_variable == 89:
    print("{} is exactly 89".format(my_variable))
else:
    print("{} is more than 89".format(my_variable))

```

Only one condition can execute

The code after the colon is “tabbed in” by 4 spaces.  
The tabbed in code is what will run if the condition is met.

Running the code on pytutor : <http://www.pythontutor.com/visualize.html>

Python 3.6

```

1 my_variable = 89
2
3 # example 1
4 if my_variable < 90:
5     print("{} is less than 90".format(my_variable))
6
7 #example 2
8 if my_variable < 89:
9     print("{} is less than 89 ".format(my_variable))
10
11 #example 3
12 if my_variable < 89:
13     print("{} is less than 89".format(my_variable))
14 else:
15     print("{} is not less than 89".format(my_variable))
16
17 #example 4
18 if my_variable < 89:
19     print("{} is less than 89".format(my_variable))
20
21

```

Print output (drag lower right corner to resize)

```

89 is less than 90
89 is not less than 89
89 is exactly 89

```

Frames Objects

Global frame  
my\_variable | 89

line that has just executed  
next line to execute  
Click a line of code to set a breakpoint; use the Back and Forward buttons to jump there.

<< First | < Back | Program terminated | Forward > | Last >>

### 3.1 Activities

1. Consider a program that asks for a user's age and then:
  - if their age 12 or less, tells them that they should be at primary school,
  - if their age is more than 12 but less than 18, they need to go to secondary school
  - if they are 18 years old, it tells them that they have become an adult
  - and if they are older than 18 tells them that they must have left school.
2. To go to university a person needs to be:
  - over the age of 20
  - **or** be 16 or older and have level 3 NCEA
  - **and** (for both conditions) the person need to be competent with English.

Create a program that given the following variables:

age= 25

ncea = False

english = True

Will give a correct response about whether a person can go to university or not.

Test with other values.

## 4 Random numbers

```
1 # import the random number library
2 import random
3
4 #random number between 0 and 1
5 print(random.random())
6
7 # random number between two integers (inclusive)
8 print(random.randint(2,5))
9
10 # random number between two integers (inclusive)
11 # but with a step value (in this case will only get multiples
12 #of 5
13 print (random.randrange(0, 101, 5))
14
15 # create a list and randomly select a name from it
16 myList = ["Anna", "Michael", "Sarah", "Jessie"]
17 print(random.choice(myList))
```

Listing 4: Python example

## 5 Program using conditions and random numbers

```
1 import random
2 lower = 4
3 higher = 15
4 my_random = random.randint(lower,higher)
5 middle_number = round((lower + higher)/2)
6 #print(middle_number)
7 #print(my_random)
8
9 print("A random number has been chosen between {} and {}".format(lower, higher))
10 print(40*"-")
11 guess = input("Enter L if you think the number is less than {0} \n"
12                 "Enter G if you think the number is more than {0} \n"
13                 "Enter E if you think the number is equal to {0} : ".format(
14                     middle_number))
15 if guess == "L" and my_random < middle_number:
16     print("you win")
17 elif guess == "G" and my_random > middle_number:
18     print("you win")
19 elif guess == "E" and my_random == middle_number:
20     print("you win")
21 elif guess != "G" and guess != "L" and guess != "E":
22     print("you have not entered an appropriate letter")
```

```
23 else:  
24     print("you lose")  
25 print("The number was {}".format(my_random))
```

Listing 5: Python example

## 6 Loops

## 7 Lists

## 8 Functions

```
1 import random
2 def addition_question():
3     a = random.randint(5,9)
4     b = random.randint(7,10)
5     my_sum = a + b
6     print("{} + {} = ".format(a,b), end="")
7     return my_sum
8
9
10 question_number = addition_question()
11 answer = int(input(" "))
12 if answer == question_number:
13     print("Correct")
```

Listing 6: Python example

```
1
2
3 my_string = "hello"
4
5 def test_function():
6     global my_string
7     my_string = "bob"
8
9 test_function()
10 print(my_string)
```

Listing 7: Python example

## 9 Dictionaries

```
1 car = {
2     "brand": "Ford",
3     "model": "Mustang",
4     "year": 1964
5 }
6
7 x = car.setdefault("model", "Bronco")
8
9 print(x)
```

```

10
11 quiz_one = {
12     "question" : "What is the capital of France? " ,
13     "answer"    : "Paris"
14 }
15 quiz_two = {
16     "question" : "What is the capital of Spain? " ,
17     "answer"    : "Madrid"
18 }
19 quiz_three = {
20     "question" : "What is the capital of Germany? " ,
21     "answer"    : "Berlin"
22 }
23 print(quiz_one.get("question"))
24
25 user_answer = input( quiz_one.get("question") )
26 if user_answer == quiz_one.get("answer"):
27     print("Correct")
28 user_answer = input( quiz_two.get("question") )
29 if user_answer == quiz_two.get("answer"):
30     print("Correct")
31 user_answer = input( quiz_three.get("question") )
32 if user_answer == quiz_three.get("answer"):
33     print("Correct")
34
35 quiz_set = [
36     {
37         "question" : "What is the capital of France? " ,
38         "answer"    : "Paris"
39     },
34     {
41         "question" : "What is the capital of Spain? " ,
42         "answer"    : "Madrid"
43     },
44     {
45         "question" : "What is the capital of Germany? " ,
46         "answer"    : "Berlin"
47     }
48 ]
49
50 for x in quiz_set:
51     user_answer = input( x.get("question") )
52     if user_answer == x.get("answer"):
53         print("Correct")

```

Listing 8: Python example

```

1 question_one = {

```

```

2     "question": "What is the capital of France?",  

3     "A": "Paris",  

4     "B": "Moscow",  

5     "C": "Shanghai",  

6     "answer": "C"  

7   }  

8 print(question_one["A"])  

9 print(question_one["B"])  

10 print("-"*30)  

11  

12 for x,y in question_one.items():  

13   if x in ["A", "B", "C"]:  

14     print("{} , {}".format(x,y))
```

Listing 9: Python example

## 10 Validation

```

1  

2 high = 10  

3 low = 1  

4 error_int = "You have not entered an integer, please try again"  

5 error_amount = ("You have not entered an appropriate amount\n"  

6                   "it must be between $1 and $10")  

7 msg = "Please choose between $1 and $10: "  

8 get_value = False  

9 while get_value == False:  

10   try:  

11     player_amount = int(input(msg))  

12   except ValueError:  

13     print(error_int)  

14     continue  

15   if player_amount < low or player_amount > high:  

16     print(error_amount)  

17     continue  

18   get_value = True
```

Listing 10: Python example

# 11 Coffee

```
1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5
6
7 def run_trial():
8     counter = 1
9     C_count = 0
10    O_count = 0
11    F_count = 0
12    E_count = 0
13    char_string = ""
14    while True:
15        my_char = random.choice(char_list)
16        char_string += my_char
17        # count number of C, O,
18        C_count = char_string.count("C")
19        O_count = char_string.count("O")
20        F_count = char_string.count("F")
21        E_count = char_string.count("E")
22        if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
23            break
24        # increment count
25        counter += 1
26        #print(char_string)
27    return len(char_string)
28
29
30 total_trials = 100
31 trials = 1
32 total_chars = 0
33 while trials <= total_trials:
34     total_chars += run_trial()
35     trials += 1
36
37
38 print(total_chars)
39 print(total_trials)
40
41 my_average = total_chars/total_trials
42 print(my_average)
```

Listing 11: Python example

```

1 import random
2
3 my_list=["c", "o", "f", "e"]
4
5 my_string = ""
6 count = 1
7 while count<100:
8     my_letter = random.choice(my_list)
9     my_string += my_letter
10    #print(my_letter)
11    if my_string.count("c")>=1 and my_string.count("o")>=1 \
12        and my_string.count("f")>=2 and my_string.count("e")>=2:
13        break
14    count += 1
15 for x in my_list:
16     print("{} * {}".format(x,my_string.count(x)))
17 print(my_string)
18 print("count is {}".format(count))

```

Listing 12: Python example

```

1 import random
2
3 my_list=["c", "o", "f", "e"]
4
5 my_string = ""
6 count = 1
7 while count<100:
8     my_letter = random.choice(my_list)
9     my_string += my_letter
10    #print(my_letter)
11    if my_string.count("c")>=1 and my_string.count("o")>=1 \
12        and my_string.count("f")>=2 and my_string.count("e")>=2:
13        break
14    count += 1
15 for x in my_list:
16     print("{} * {}".format(x,my_string.count(x)))
17 print(my_string)
18 print("count is {}".format(count))

```

Listing 13: Python example

```

1 import random
2
3 my_list=["c", "o", "f", "e"]
4
5 string_list = []
6

```

```

7 counter = 0
8 while counter < 1000:
9     my_string = ""
10    count = 1
11    while True:
12        my_letter = random.choice(my_list)
13        my_string += my_letter
14        #print(my_letter)
15        if my_string.count("c")>=1 and my_string.count("o")>=1 \
16            and my_string.count("f")>=2 and my_string.count("e")>=2:
17            break
18        count += 1
19        string_list.append(my_string)
20        counter += 1
21        # print(my_string)
22 #print(string_list)
23 string_list.sort(key=len)
24 print(string_list[0])
25 print(string_list[- 1])
26
27 total_times = 0
28 number_trials = 0
29 for x in string_list:
30     total_times += len(x)
31     number_trials += 1
32 print(total_times)
33 print(number_trials)

```

Listing 14: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5
6 chars_in_six = 0
7
8 C_count = 0
9 O_count = 0
10 F_count = 0
11 E_count = 0
12
13
14 trials = 0
15 trial_total = 100000
16 while trials < trial_total:
17     counter = 0
18     char_string = ""

```

```

19     while counter < 6:
20         my_char = random.choice(char_list)
21         char_string += my_char
22         # count number of C, O,
23         C_count = char_string.count("C")
24         O_count = char_string.count("O")
25         F_count = char_string.count("F")
26         E_count = char_string.count("E")
27         if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
28             chars_in_six +=1
29             # increment count
30             counter += 1
31         #print(char_string)
32         trials += 1
33
34 print(chars_in_six/trial_total)
35 #print(char_string.count("C"))

```

Listing 15: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5 # empty list
6 string_list = []
7
8
9 #-----
10 def make_coffee():
11     C_count = 0
12     O_count = 0
13     F_count = 0
14     E_count = 0
15
16     char_string = ""
17     while True:
18         my_char = random.choice(char_list)
19         char_string += my_char
20         # count number of C, O,
21         C_count = char_string.count("C")
22         O_count = char_string.count("O")
23         F_count = char_string.count("F")
24         E_count = char_string.count("E")
25         if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
26             string_list.append(char_string)
27             break
28             # increment count

```

```

29
30
31     #print(char_string)
32     #print( len(char_string) )
33     return len(char_string)
34 #-----
35
36
37 total_trials = 1000000
38 trials = 1
39
40 total_string_lengths = 0
41
42 while trials <= total_trials:
43     my_number = make_coffee()
44     total_string_lengths += my_number
45     #print(my_number)
46     trials +=1
47 #print(total_string_lengths)
48 average = round(total_string_lengths / total_trials)
49 print("On average you will need {} visits".format(average))
50 #print(string_list)
51
52 string_list.sort(key=len)
53 #print(string_list)
54 #print first item (thing)
55 print(string_list[0])
56 #print last item
57 print(string_list[total_trials-1])
58 print( len( string_list[total_trials-1] ) )
59
60
61
62
63
64
65
66
67
68
69 #print(string_list)
70
71
72
73 #string_list.sort(key = len)

```

Listing 16: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5 string_list = []
6
7
8 def run_trial():
9     counter = 1
10    C_count = 0
11    O_count = 0
12    F_count = 0
13    E_count = 0
14    char_string = ""
15    while True:
16        my_char = random.choice(char_list)
17        char_string += my_char
18        # count number of C, O,
19        C_count = char_string.count("C")
20        O_count = char_string.count("O")
21        F_count = char_string.count("F")
22        E_count = char_string.count("E")
23        if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
24            string_list.append(char_string)
25            break
26        # increment count
27        counter += 1
28    #print(char_string)
29    return len(char_string)
30
31
32
33
34 total_trials = 100
35 trials = 1
36 total_chars = 0
37 while trials <= total_trials:
38     total_chars += run_trial()
39     trials += 1
40
41
42 print(total_chars)
43 print(total_trials)
44 my_average = total_chars/total_trials
45 print(my_average)
46
47 #print(string_list)

```

```

48 string_list.sort(key = len)
49 #print(string_list)
50 print(string_list[0])
51 print(string_list[-1])
52 print(len(string_list[0]))
53 print(len(string_list[-1]))
54
55 my_dict = {}
56 for i in range(len(string_list[0]), len(string_list[-1])+1):
57     my_dict[i]=0
58     #print(i)
59
60 #print(my_dict)
61 for x in string_list:
62     my_dict[len(x)]+=1
63     #print(x)
64 print(my_dict)
65 my_sum = 0
66 for x in my_dict:
67     my_sum += my_dict[x]
68 print(my_sum)
69 # dictionary not the same lenght as string list
70 for x,y in my_dict.items():
71     temp = int(y)
72     temp = temp/total_trials
73     print(temp)
74     my_dict[x] = temp
75 print(my_dict)

```

Listing 17: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5
6 chars_in_seven = 0
7
8 C_count = 0
9 O_count = 0
10 F_count = 0
11 E_count = 0
12
13
14 trials = 0
15 trial_total = 1000000
16 while trials < trial_total:
17     counter = 1

```

```

18     char_string = ""
19     while counter <= 7:
20         my_char = random.choice(char_list)
21         char_string += my_char
22         # count number of C, O, F, E
23         C_count = char_string.count("C")
24         O_count = char_string.count("O")
25         F_count = char_string.count("F")
26         E_count = char_string.count("E")
27
28         # increment count
29         counter += 1
30     # test string at end
31     if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
32         chars_in_seven +=1
33     #print(char_string)
34     trials += 1
35
36 print(chars_in_seven/trial_total)
37 #print(char_string.count("C"))

```

Listing 18: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5
6 chars_in_eight = 0
7
8 C_count = 0
9 O_count = 0
10 F_count = 0
11 E_count = 0
12
13
14 trials = 0
15 trial_total = 1000000
16 while trials < trial_total:
17     counter = 1
18     char_string = ""
19     while counter <= 8:
20         my_char = random.choice(char_list)
21         char_string += my_char
22         # count number of C, O, F, E
23         C_count = char_string.count("C")
24         O_count = char_string.count("O")
25         F_count = char_string.count("F")

```

```

26     E_count = char_string.count("E")
27
28     # increment count
29     counter += 1
30
31     # test string at end
32     if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
33         chars_in_eight +=1
34     #print(char_string)
35
36 trials += 1
37
38 print(chars_in_eight/trial_total)
39 #print(char_string.count("C"))

```

Listing 19: Python example

```

1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5
6
7 C_count = 0
8 O_count = 0
9 F_count = 0
10 E_count = 0
11
12 char_string = ""
13 while True:
14     my_char = random.choice(char_list)
15     char_string += my_char
16     # count number of C, O,
17     C_count = char_string.count("C")
18     O_count = char_string.count("O")
19     F_count = char_string.count("F")
20     E_count = char_string.count("E")
21     if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
22         break
23     # increment count
24
25
26 print(char_string)
27 print( len(char_string) )
28
29 #print(char_string.count("C"))

```

Listing 20: Python example

```
1 import random
```

```

2
3 char_list = ["C", "O", "F", "E"]
4
5
6
7
8 #-----
9 def make_coffee():
10     C_count = 0
11     O_count = 0
12     F_count = 0
13     E_count = 0
14
15     char_string = ""
16     while True:
17         my_char = random.choice(char_list)
18         char_string += my_char
19         # count number of C, O,
20         C_count = char_string.count("C")
21         O_count = char_string.count("O")
22         F_count = char_string.count("F")
23         E_count = char_string.count("E")
24         if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
25             break
26         # increment count
27
28
29     #print(char_string)
30     #print( len(char_string) )
31     return len(char_string)
32 #-----
33
34
35 total_trials = 100
36 trials = 1
37
38 total_string_lengths = 0
39
40 while trials <= total_trials:
41     my_number = make_coffee()
42     total_string_lengths += my_number
43     #print(my_number)
44     trials +=1
45 #print(total_string_lengths)
46 average = round(total_string_lengths / total_trials)

```

```
47 print("On average you will need {} visits".format(average))
```

Listing 21: Python example

```
1 import random
2
3 char_list = ["C", "O", "F", "E"]
4
5 string_list = []
6
7
8 def run_trial():
9     counter = 1
10    C_count = 0
11    O_count = 0
12    F_count = 0
13    E_count = 0
14    char_string = ""
15    while True:
16        my_char = random.choice(char_list)
17        char_string += my_char
18        # count number of C, O,
19        C_count = char_string.count("C")
20        O_count = char_string.count("O")
21        F_count = char_string.count("F")
22        E_count = char_string.count("E")
23        if C_count >= 1 and O_count >= 1 and F_count >= 2 and E_count >= 2:
24            string_list.append(char_string)
25            break
26        # increment count
27        counter += 1
28    #print(char_string)
29    return len(char_string)
30
31
32
33
34 total_trials = 1000000
35 trials = 1
36 total_chars = 0
37 while trials <= total_trials:
38     total_chars += run_trial()
39     trials += 1
40
41
42 print(total_chars)
43 print(total_trials)
44 my_average = total_chars/total_trials
```

```

45 print(my_average)
46
47 #print(string_list)
48 string_list.sort(key = len)
49 #print(string_list)
50 print(string_list[0])
51 print(string_list[-1])
52 print(len(string_list[0]))
53 print(len(string_list[-1]))
54
55 my_dict = {}
56 for i in range(len(string_list[0]), len(string_list[-1])+1):
57     my_dict[i]=0
58     #print(i)
59
60 #print(my_dict)
61 for x in string_list:
62     my_dict[len(x)]+=1
63
64 print(my_dict)
65 my_sum = 0
66 for x in my_dict:
67     my_sum += my_dict[x]
68 print(my_sum)
69 # dictionary not the same lenght as string list
70 for x,y in my_dict.items():
71     temp = int(y)
72     temp = temp/total_trials
73     my_dict[x]= temp
74 print(my_dict)
75
76 my_file = open("distribution_next.txt","a")
77 for x,y in my_dict.items():
78     my_string = "{},{:.6f}\n".format(x,y)
79     my_file.write(my_string)
80
81 #my_file.write("hello")
82 #my_file.write("\n")
83 #my_file.write("goodbye")
84 my_file.close()
85 #my_file = open("MyFile.txt","w")
86 #my_file.write("")
87 #my_file.close()

```

Listing 22: Python example

## 12 Tests

```
1 # roll a dice
2 import random
3 two_d_list = []
4
5
6
7 def searchfor(n):
8     if len(two_d_list)== 0:
9         two_d_list.append([n,1])
10        return
11    added = False
12    for sublist in two_d_list:
13        if sublist[0]==n:
14            sublist[1] += 1
15            added = True
16            break
17    if added == False:
18        two_d_list.append([n,1])
19
20 for i in range(0,1500):
21     diceroll = random.randint(1,6)+random.randint(1,6)+random.randint(1,6)
22     searchfor(diceroll)
23     #print(diceroll)
24
25
26 two_d_list.sort()
27 print(two_d_list)
```

Listing 23: Dice Roll

```
1 guess_list = [45,78,90,21,7]
2
3 guess=int(input("Enter your guess"))
4
5
6 while guess in guess_list:
7     guess = int(input("You have already tried this number,\n
8 have another guess"))
9 guess_list.append(guess)
10 print(guess_list)
```

Listing 24: Python example

```
1 import time
2 start= time.time()
3 t= time.localtime()
```

```

4 print(t)
5 print(start)
6
7
8
9
10 word = "halloween"
11 char_list = list(word)
12 print(char_list)
13 char_list_chosen=["l", "w", "e"]
14 outstring = ""
15 for x in char_list:
16     if x in char_list_chosen:
17         outstring += " "+x+" "
18     else:
19         outstring += " _ "
20 print(outstring)
21
22
23
24
25
26
27 question_one = {
28     "question": "What is the capital of France?",
29     "A": "Paris",
30     "B": "Moscow",
31     "C": "Shanghai",
32     "answer": "C"
33 }
34
35 print(question_one["question"])
36 for x,y in question_one.items():
37     if x != "question" and x != "answer":
38         print("{}) {}".format(x,y))
39
40 for x,y in question_one.items():
41     if len(x)== 1:
42         print("{}) {}".format(x,y))
43
44 answer = input("Enter your choice ")
45 if answer == question_one["answer"]:
46     print(" Brilliant ")
47 end=time.time()
48 print(end)
49 print(end-start)
50 count = 0

```

```

51 while count < 3:
52     temp={}
53     temp["english"] = input("Enter your english word")
54     temp["thai"] = input("Enter your thai word")
55     print(temp)
56     count +=1

```

Listing 25: Python example

```

1 my_num=0.0000067
2 print("{:.6f}".format(my_num) )

```

Listing 26: Python example

```

1 word = "halloween"
2
3 def pic(c):
4     if c >= 1:
5         print(" "*30+" - - - -")
6     if c >=2:
7         print(" "*30+" | | ")
8     if c >=3:
9         print(" "*30+" ( ) | ")
10    if c >= 4:
11        print(" "*30+" | | ")
12    if c >= 5:
13        print(" "*30+" --- | ")
14        print(" "*30+" | | ")
15        print(" "*30+" / \ | ")
16        print(" "*30+" _ _ | ")
17 pic(5)
18 char_list = list(word)
19 char_individual_list=[]
20 char_list_chosen=[]
21
22 for x in char_list:
23     if x not in char_individual_list:
24         char_individual_list.append(x)
25
26 wrong_count = 0
27 while len(char_individual_list)>0:
28     outstring = ""
29     for x in char_list:
30         if x in char_list_chosen:
31             outstring += " "+x+" "
32         else:
33             outstring += " _ "
34     print("{}\n\n".format(outstring))

```

```

35     char = input("guess a letter: ")
36     if char in char_individual_list:
37         char_list_chosen.append(char)
38         char_individual_list.remove(char)
39     else:
40         wrong_count += 1
41         pic(wrong_count)
42     if len(char_individual_list) == 0:
43         print("you got it")
44
45
46 outstring = ""
47 for x in char_list:
48     if x in char_list_chosen:
49         outstring += " "+x+" "
50     else:
51         outstring += " _ "
52 print(outstring)

```

Listing 27: Python example

## 13 Strings

```

1 my_string="adkadkkdjk jkjdhkdh"
2 my_split=my_string.split("a")
3 print(my_split)
4 my_list=list(my_string)
5 print(my_list)

```

Listing 28: Python example

```

1 my_string = "woollooomooloo"
2
3 my_letters=[]
4 count_letters = []
5
6 for x in my_string:
7     if x not in my_letters:
8         my_letters.append(x)
9         count_letters.append(my_string.count(x))
10
11 print(my_letters)
12 print(count_letters)

```

Listing 29: Python example

## 14 Objects

```

1 class Shark:
2
3     def __init__(self, name, age):
4         self.name = name
5         self.age = age
6         """ constructor """
7         print("This is the constructor")
8
9     def swim(self):
10        """ what the shark is doing"""
11        print(self.name+" is swimming")
12
13    def be_awesome(self):
14        """ what the shark is being"""
15        print(self.name+" is being awesome")
16
17    def _age(self):
18        """ how old is it """
19        print(self.name + " is "+str(self.age)+" years old")
20
21
22 def main():
23     sammy = Shark("Sammy", 5)
24     stevie = Shark("Stevie", 3)
25     sammy.swim()
26     stevie.be_awesome()
27     sammy._age()
28
29 if __name__ == "__main__":
30     main()

```

Listing 30: Python example

```

1 class Store:
2
3     def __init__(self):
4         self.colour = "rgb(0,0,0,0)"
5
6
7
8
9
10 class Shark:
11
12     def __init__(self, name, age):
13         self.name = name
14         self.age = age
15         """ constructor """

```

```

16     store.colour = "rgb(255,255,255,1)"
17     print("This is the constructor")
18
19     def swim(self):
20         """ what the shark is doing"""
21         print(self.name+" is swimming")
22
23     def be_awesome(self):
24         """ what the shark is being"""
25         print(self.name+" is being awesome")
26
27     def _age(self):
28         """ how old is it """
29         print(self.name + " is "+str(self.age)+" years old")
30
31 store = Store()
32 shark = Shark("sam", 10)
33 print(store.colour)

```

Listing 31: Python example

```

1 class Question:
2
3     def __init__(self, question, answer, score):
4
5         self.question = question
6         self.answer = answer
7         self.score = score
8
9         self.user_answer = ""
10        self.user_score = 0
11        self.response_to_user = ""
12
13    def make_question(self):
14        msg = self.question
15        self.user_answer = input(msg)
16        if self.user_answer == self.answer:
17            self.response_to_user = "Correct"
18            self.user_score = self.score
19        else:
20            self.response_to_user = "Incorrect"
21
22    def feedback(self):
23        strOne = "For the question: "+self.question+"\n"
24        strTwo = "You answered: "+self.user_answer+"\n"
25        strThree = "Your answer was "+self.response_to_user
26        print(strOne + strTwo + strThree)

```

```

28 #q_one = Question("What is the capital of France?", "Paris", 10)
29 #q_one.make_question()
30 #q_one.feedback()
31 q_list = []
32 q_list.append(Question("What is the capital of France?", "Paris", 10))
33 q_list.append(Question("What is the capital of Belgium?", "Brussels", 10))
34 q_list.append(Question("What is the capital of Holland?", "Amstredam", 10))
35
36 count = 0
37 while count < len(q_list):
38     q_list[count].make_question()
39     count += 1
40
41 count = 0
42 while count < len(q_list):
43     q_list[count].feedback()
44     count += 1

```

Listing 32: Python example