Assignment 2

Key information (front page)

How will I be assessed

60% team mark (common): based on submission

- Approximately 20% of the team mark from test cases.
- Approximately 80% of the team mark from review by your teacher as per rubric (Appendix 1).

35% based on interview (individual)

Details To Be Announced.

5% for submitting feedback to team members (individual)

Details To Be Announced.

Review of your work by your tutor

Your work will be looked at **overall** (not necessarily every single task) and this will align with the rubric included.

This is to find evidence you are able to use the different techniques you've learnt about so far in the unit (e.g. do you understand how to meaningfully design a conditional to achieve some broader goal)

Maximising your marks from tutor review

You should aim to ensure your program includes the elements in the rubric (next page).

To ensure this is the best representation of your ability, you should ensure you contribute to tasks in workshops and applied classes so that the **feedback** your group receives will be **meaningful** for you and will guide you in improving proficiency.

Academic integrity

How will this be checked?

Your code will be compared against every other students' work in the unit.

You should also assume that anything you are able to google can be easily found by the teaching team and compared against your work.

You will also be **individually** interviewed on your understanding of different components of the program, whatever your contribution to a particular component might be. You are expected to understand all parts of the program equally well. Make sure you work as a team to ensure that.

How can I avoid academic integrity issues?

- Never copy code from anywhere. If you learn something useful online, rewrite it from scratch. It's also the best way to make sure you have understood it. If you're concerned you may cause an academic integrity case by copying something on the internet, the easiest way to avoid this is to not search anything too specific. Once you read a solution, it is very hard to forget it.
- If a fellow student asks you for a solution to a question, try instead to figure out what it is that they do not understand about the unit's content that prevents them from finding the solution themselves. Give a man a fish, and you feed him for a day. Teach a man to fish, and you feed him for a lifetime. Also, remember that giving your solution is just as much of an Academic Integrity breach as receiving it!
- You may find yourself in a situation where you feel like you physically cannot submit the assignment on time. Remember that you can submit an extension request (see Moodle AU or Moodle MA), and you can seek help (see Moodle AU or Moodle MA). If nothing works, remember that failures are part of the learning journey. And what is more important to you, an assignment mark or acting honourably?

Automated testing (and testing your own code)

We have included some test files for each of the python files we expect you to produce (in the Assignment 2 scaffold workspace, see details of accessing in teamwork section).

To run these tests yourself you will need to go to the console and run

```
python test_XXX.py
```

The results you see in the console will be the results of running these tests

We will use the test cases for the 20% of your marks coming from automated tests; do not modify these files as they will simply be overwritten by your TA when it comes to marking your work.

Checking that you have the right test files and that they are not modified (change 2)

One way to check that you are using the right test files is to type the command `md5sum test_*` in the terminal of your workspace. If you have the correct test files, you should see exactly this:

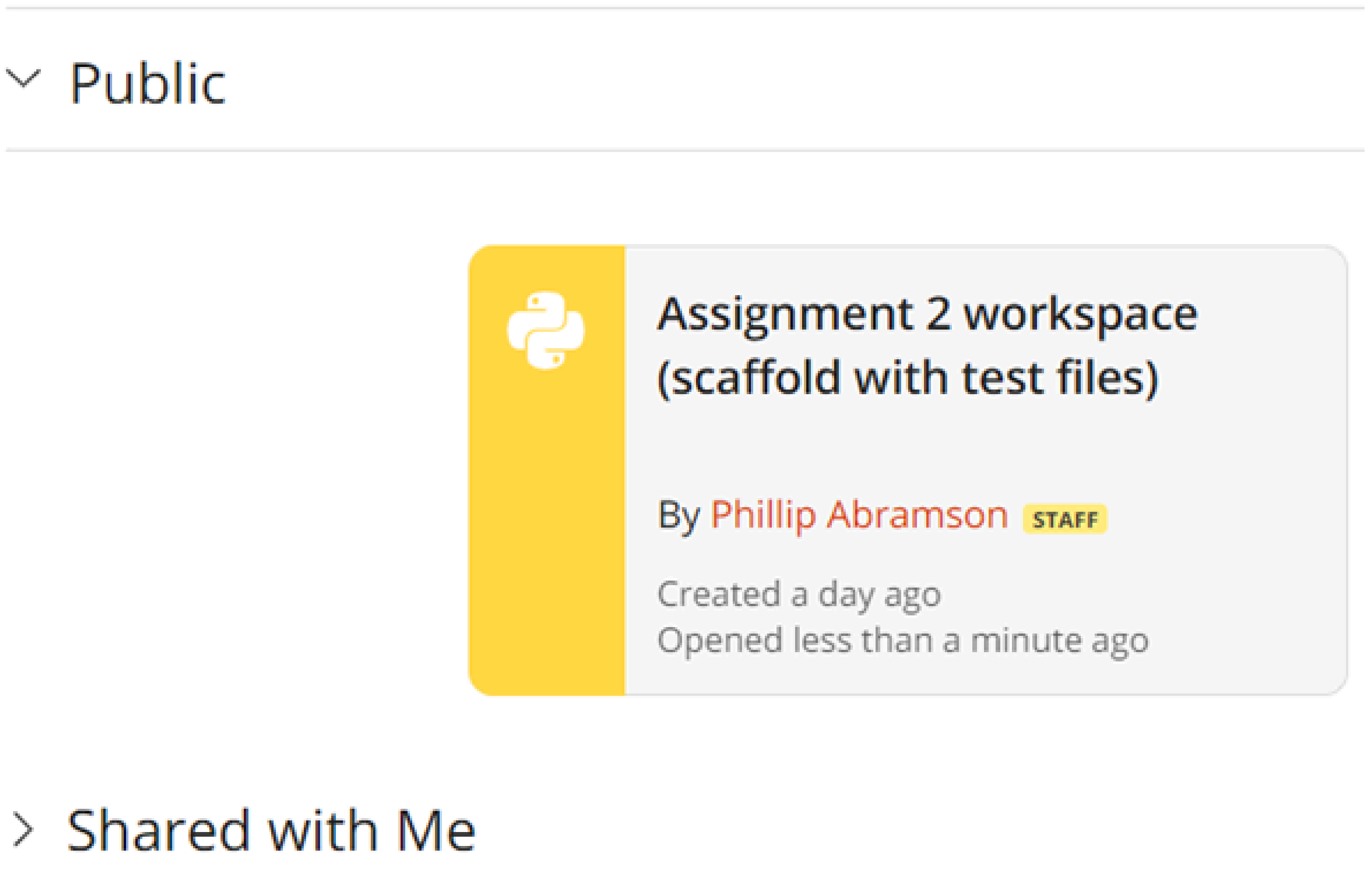
```
b4f092419b8aeef21435b055842d1e3d  test_classify_cooking_for_tip.py
2d4caf25134ea325b32d4b0c17b0a39b  test_customer_selection.py
fa08c5fe151826d4af4baa87a9b0f553  test_get_meal.py
4c528bd57004b3564f7b6dd196f72466  test_get_menu.py
7db118c9dd1d8bbf94072859b84ec2fe  test_restaurant_simulator.py
```

If you don't, try to grab a fresh copy of the test files from the workspace and try again. Please let us know if that does not work.

Assignment Rubric

Teamwork

In this assignment, you are expected to work in teams of 4. You will need to meet regularly with your team and collaborate using the workspace in Ed. There will be a scaffold workspace in Ed for you to fork using the button in the top right as shown below.



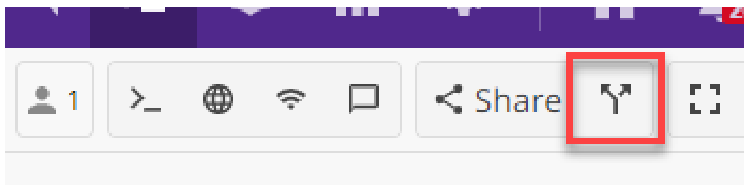
Public

Assignment 2 workspace (scaffold with test files)

By Phillip Abramson STAFF

Created a day ago
Opened less than a minute ago

Shared with Me



Make sure to share your workspace with you teammates. We recommend you use a single workspace, but you can use more if that works better for your team.

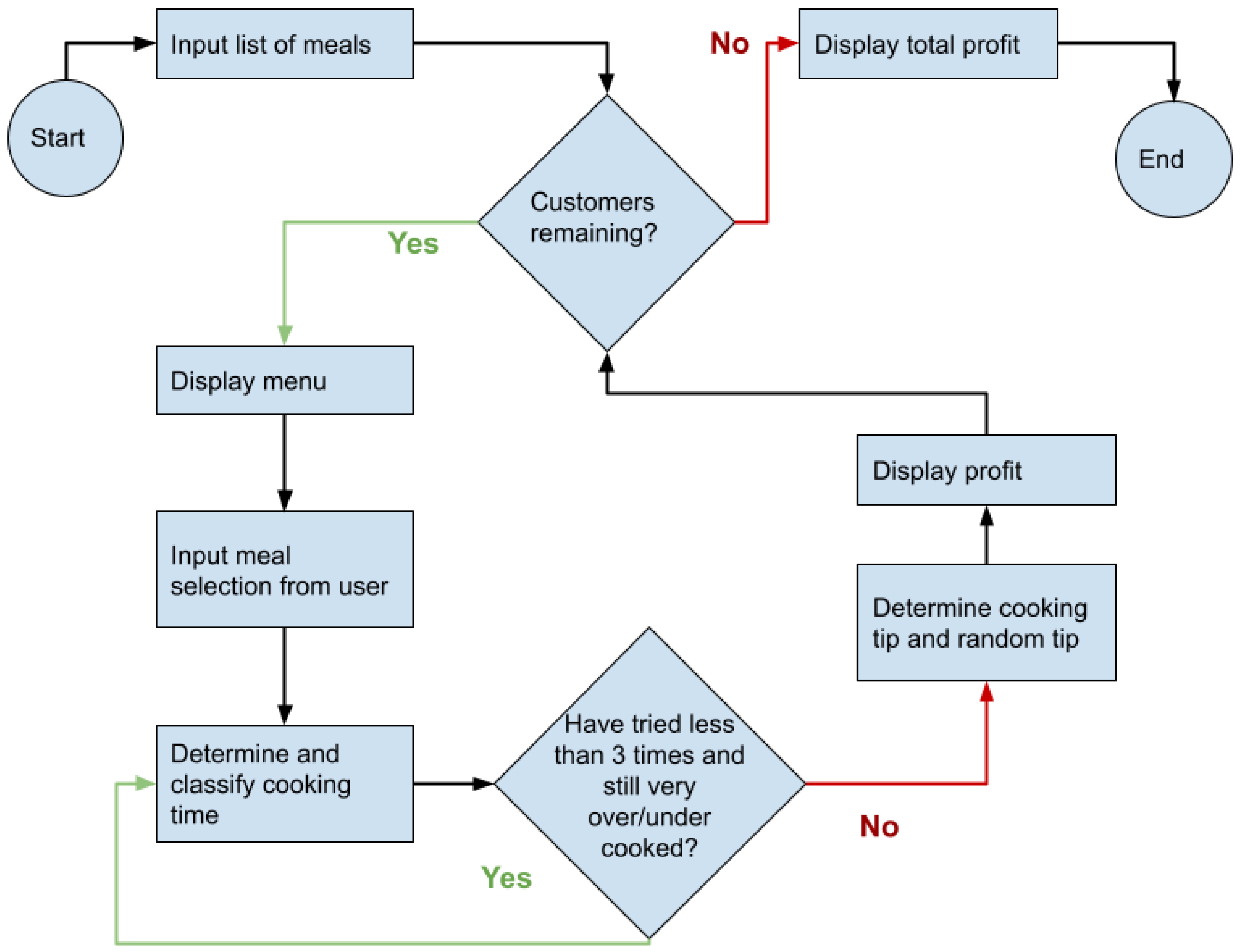
Once you know the tutor who will mark you, add them to your workspace prior to the interview. They will be able to see who has contributed over the course of the assessment period through the replay option.

After you submit the assignment, you will be asked to complete a peer review of one another’s contributions to the team, the submission of which will contribute to your individual mark.

Description of Task

Overview of Assignment

Note: This represents the order they will **occur** in in the final program **not** the order we **recommend** you attempt them in (this is given later).


Start
Input list of meals
Customers remaining?
No
Display total profit
End
Yes
Display menu
Input meal selection from user
Determine and classify cooking time
Have tried less than 3 times and still very over/under cooked?
Yes
No
Determine cooking tip and random tip
Display profit

Requirement 1 (R.1) Input list of meals.

- A set of meals will be given as input to the program.
- These will need to be read, interpreted and represented as lists of dictionaries.

Requirement 2 (R.2) Display Menu.

- You will have a list of meals to display to the user.

Requirement 3 (R.3) User selects a meal option.

- Input is taken from the user to choose from the menu displayed.

- Appropriate validation should be performed.

Requirement 4 (R.4) Use a normally distributed random number to simulate cooking time.

- Each meal will have a mean (or average) cooking time which in this case we assume also represents a perfectly cooked meal.
- Meals also have a standard deviation of cooking time (essentially how much variability is there in how long meals they are cooked for.
- These will be used with the random.gauss function to estimate the time spent cooking this specific customer's meal.

Requirement 5 (R.5) Classify the cooking time and determine the tip a customer is willing to pay on this basis.

- Well cooked meals get an additional tip as an additional percentage of the base selling price (e.g. +10% tip on $50 meal = $55)
- Slightly over or under cooked meals do not get this tip.
- Very over or under cooked meals are sent back to the kitchen.

Requirement 6 (R.6) Retry cooking a meal up to twice more if very over or undercooked.

- As per step 4 a meal may be sent back.
- The cooks in the kitchen are willing to try up to three times to cook a meal before they give up -- we can assume in this case the customer leaves.

Requirement 7 (R.7) Calculate any additional tip paid by the customer.

- There is a random chance a customer will add an additional tip for any meal that is **not sent back**.
    - This is as a percent of the base meal price (e.g. +10% cooking tip and +5% random tip on a $50 meal = 50 + 5 + 2.5 = $57.50)
- This is determined by using the random.random function against a particular chance.
- In some cases a customer will be difficult and the restaurant offers a partial discount (represented by a negative tip).

Requirement 8 (R.8) Calculate the profit for this meal.

- This will include the tips in steps 4 and 6 as well as the costs to produce the meal (include reproducing as per step 5).
- This will be displayed to the user in stages (explained later).

Requirement 9 (R.9) Repeat R.1-8 for each customer in the restaurant and determine overall profit.

- This will require keeping a running tab of profit / costs.
- At the end the final profit for the day will be displayed.

Important: Do's and Don'ts

Do

- ✓ Add your teammates to your ed workspace.
- ✓ Run your work regularly and test with different scenarios.
- ✓ Use the function names and arguments given when writing your code.
- ✓ Program, review and discuss your code with your teammates.
- ✓ Add the TA marking your assignment to your ed workspace once you know who they will be.
- ✓ Use `input()`, not `input('')`, to read from the keyboard without a prompt. (this is change 3)

Don’t

- ✗ Import any libraries or packages aside from random (you won’t need them).
- ✗ Change the test files provided. We will check.
- ✗ Work or ask other members of your team to work in isolation.

Changelog (6 changes)

Changes made to the instructions will be reported here. All times are Melbourne time.

Change 6 - 31/02/2022 at 09:55

Fixed "**R5.** `classify_cook_time`" instruction (see this slide, under one of the expand buttons). Thank you Vincent for your post!

Change 5 - 29/03/2022 at 12:05

Changed the test case for `classify_cook_time` in the file *test_restaurant_simulator.py*, updated the slide and the md5sum of the test. Thanks to those who raised the issue here and here.

Change 4 -28/03/2022 at 17:00

Added a slide on Moodle submission.

Change 3 - 25/03/2022 at 15:50

Added a Do about using `input()` in this slide. Thanks Sean for raising the issue here.

Change 2 - 24/03/2022 at 16:15

Added information to the slide on automated testing on how to check that the test files are correct.

Change 1 - 24/03/2022 at 16:10

Changed the file *test_get_menu.py* and updated slide R.2. Thanks Tam for pointing out the error in this thread.

Submit your work on Moodle

When you have completed your work, download it from your workspace and upload it on the Moodle "Assignment 2 submission" box on the assessment page. One submission per team!

This will be used to make sure any accidental changes to the workspace after the due date will not incur late penalties.

(this is change 4)

R.1. Interpreting a list of meals

This task will take input from the user representing meal information and store these internally.

Supporting information

Meal information consists of the following information:

- `name`: the name of the meal (e.g. Lamb Madras),
- `sell_for`: the price the meal is typically sold for (e.g. $25.49),
- `cost_to_make`: the total cost in terms of ingredients, labour, electricity, etc. to produce the meal,
- `cook_time`: the time (in minutes) **required** to cook this meal well (this is **also** the average cooking time of restaurant staff for this meal),
- `cook_time_stdev`: typical variation (in minutes) of the cooking time of restaurant staff for this meal, specifically the standard deviation.

What is standard deviation?

You don't need to know exactly what this is, beyond that

- It measures how spread out data is compared to the average (a.k.a. mean),
    - Essentially the distance that data might be found from the average on a graph,
- Are always positive numbers.

You can read more about standard deviation online, however it is not necessary for your understanding of this task.

Your task

Write a program called ***get_meal.py*** which

- Takes as input a series of meal details (as described above), one meal per line,
- Encodes each numbers as a `float`,
- Stores each meal and its details into its own dictionary (assume the order matches what's given in the supporting information),
- Retains the entire set of meals as a list of dictionaries. The list must be named `options`,
- Stops reading on receiving a period (i.e. the character `'.'`),
- Ends the program by running `print(options)`.

For testing purposes, ensure your dictionaries use the property names given in the supporting information, in the same order that we provide (i.e. `name`, `sell_for`, ..., `cook_time_stdev`*).*

You will need to use

Python dictionaries (i.e. `dict`)

You may find these helpful

String split operation: https://www.w3schools.com/python/ref_string_split.asp

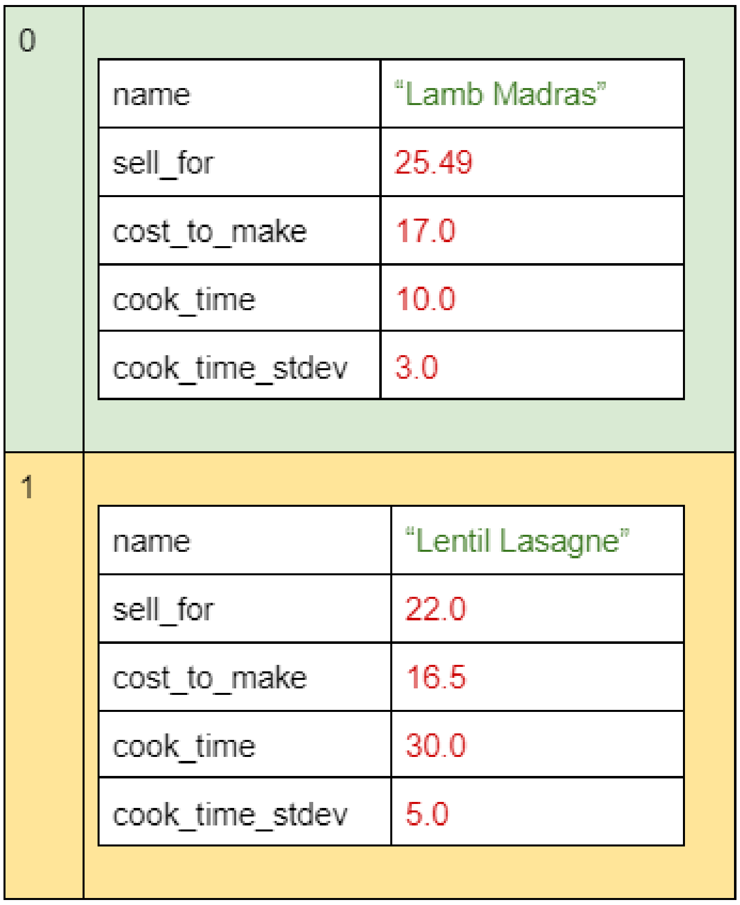
Sample input and result

Input 1

```
Lamb Madras,25.49,17,10,3
Lentil Lasagne,22,16.5,30,5
.
```

State of the variable `options`

(In this example, the list *options* contains two items, each of them a dictionary representing a meal and its details, as shown below).



0	name	"Lamb Madras"
	sell_for	25.49
	cost_to_make	17.0
	cook_time	10.0
	cook_time_stdev	3.0
1	name	"Lentil Lasagne"
	sell_for	22.0
	cost_to_make	16.5
	cook_time	30.0
	cook_time_stdev	5.0

Output 1

```
[{'name': 'Lamb Madras', 'sell_for': 25.49, 'cost_to_make': 17.0, 'cook_time': 10.0, 'cook_time_stdev': 3.0}, {'name': 'Lentil Lasagne', 'sell_for': 22.0, 'cost_to_make': 16.5, 'cook_time': 30.0, 'cook_time_stdev': 5.0}]
```

Note: the first element of the list is bolded to help differentiate the meals (this will not be done in the output you produce)

Input 2

```
.
```

State of the variable `options`

`options` is empty

Output 2

```
[]
```

R.2. Display Menu

Your task

Change 1 - Thursday 24/03/2022 9:30am Melbourne time:
We have updated the test file *test_get_menu.py* (used for R2) in the original workspace. You will need to grab the new file and replace your own (if don't already have the most recent one).

Write a program called ***get_menu.py*** which

- Use the program you wrote for R.1. to receive a list of menu items,
    - You should copy-paste the relevant code from this into *get_menu.py* to ensure the automated tests run correctly,
- If you receive a period (`'.'`) with no meal information at all, use the default menu item as follows.

```
Budda Bowl (vg),25,20,10,3
Eye Fillet Steak,55,25,7,1
Spaghetti Bolognese,30,22,40,5
Pad Thai (seafood),22,17,30,1
```

- Print off a menu for the user listing the meals in the order they were inputted and matching the format below.

[number]. Name:[meal name] Sells:$[selling price] Costs:$[cost to make] Takes:[cook time] mins

Note: do **not** import *get_meal.py*. Copy-paste the code you need from it. Make sure you comment/delete the `print(options)` line to obtain the correct output and avoid confusing the automated tester.

You will need to use

Your program for R.1

You may find this helpful

The python `+` operation for joining strings

We allow you to explore and use other approaches for formatting strings in Python (e.g. f-strings, string.format, etc, see for example https://realpython.com/python-f-strings/).

Sample input and output

Input 1

```
Lentil Lasagne,22,16.5,30,5
Lamb Madras,25.49,17,10,3
.
```

Output 1

```
1. Name:Lentil Lasagne Sells:$22.0 Costs:$16.5 Takes:30.0 mins
2. Name:Lamb Madras Sells:$25.49 Costs:$17.0 Takes:10.0 mins
```

Input 2

```
.
```

Output 2

```
1. Name:Budda Bowl (vg) Sells:$25.0 Costs:$20.0 Takes:10.0 mins
2. Name:Eye Fillet Steak Sells:$55.0 Costs:$25.0 Takes:7.0 mins
3. Name:Spaghetti Bolognese Sells:$30.0 Costs:$22.0 Takes:40.0 mins
4. Name:Pad Thai (seafood) Sells:$22.0 Costs:$17.0 Takes:30.0 mins
```

R.3. User selects a meal option

Your task

Write a program called ***customer_selection.py*** which

- Uses your program for R.2. to receive a list of menu items (as with R.2. you can assume an input of the character '.' with no meal information should use the default values given there)
  - Once again, you will need to copy-paste the relevant code from this into customer_selection.py to ensure the automated tests run correctly
- Allows the user to enter a meal selection number (which matches the number shown in the menu)
- Appropriate validation should be performed on user response
  - After a valid choice the program should print "now cooking [`name`]"
  - Otherwise it should print "invalid choice"

You will need to use

Your program for R.1 or R.2

Sample input and output

Input 1

```
.
4
```

Output 1

```
now cooking Pad Thai (seafood)
```

Input 2

```
.
5
```

Output 2

```
invalid choice
```

Input 3

```
.
Spaghetti Bolognese
```

Output 3

```
invalid choice
```

(Only the number should be used to select a meal)

R.5. Classify cooking time and determine the tip that the customer pays

Supporting information

The categories / classification of cooking time is given in the table below

Here we will refer to the average (and also correct) cooking time as **μ**, the standard deviation of this as **σ**, and the actual cooking time as **T**.

Cases

T is less than **μ** - 2**σ**

▶ Expand

T is between (or includes) **μ - 2σ and μ - σ**

▶ Expand

T is between **μ - σ and μ + σ** (excluding these values themselves)

▶ Expand

T is between (or includes) **μ + σ and μ + 2σ**

▶ Expand

T is greater than **μ** + 2**σ**

▶ Expand

Note: This formula technically allows negative cooking times, you do not need to consider the consequences of this

Your task

Write a program called **classify_cooking_for_tip.py** which

- Uses your program for R.1. to receive a list of menu items (as with R.2. you can assume an input of "." with no meal information should use the default values given there)
- Takes as input a meal selection number (assume this is a valid selection), actual cooking time in the form

  “[meal number],[actual cooking time]”
- Prints out the cooking category (see supporting information) and corresponding tip in the form “[name] was [classification] and cooking tip was [final cooking tip]%”

You will need to use

Your program for R.1 or R.2

Sample input and output

Input 1

```
.
2,40
```

Output 1

```
Eye Fillet Steak was very overcooked and cooking tip was -100%
```

Input 2

```
.
1,5
```

Output 2

```
Budda Bowl (vg) was slightly undercooked and cooking tip was 0%
```

Input 3

```
.
4,30.5
```

Output 3

```
Pad Thai (seafood) was well cooked and cooking tip was 10.0%
```

R.4. and R.7. Using uniform and normally distributed random numbers for cooking time and additional tip

Note: Due to the nature of this task, automated testing will not be performed.

Supporting information

Customer-selected tip

There are three scenarios for this as shown in the table below for the imaginary customers Karen, Kris and Kai. Your program must produce these tip variations at the frequencies provided.

Scenarios

Customer given a discount / partial refund

▶ Expand

No change

▶ Expand

Tip paid

▶ Expand

random.random()

This is the random module's **uniform** random number generator. It will generate a floating point number from 0.0 to 1.0 where every unique number in the range has an equal chance of occurring (e.g. 0.501 is just as likely as 0.999).

Based on this you could assume that the chance of seeing a number within a given 25% of this range would also be 25% (e.g. a 1 in 4 chance of seeing a result between 0.5 and 0.75).

It is called using the `random()` function of the `random` module as shown below.

Sample use

```
import random
print(random.random())
print(random.random())
print(random.random())
```

random.gauss()

This is the random module’s **normally distributed** random number generator. It will take an average/mean (e.g. the average cooking time) and a standard deviation (e.g. the standard deviation of the cooking time) and come back with a number some distance from the mean based on that standard deviation. A smaller standard deviation means more results will likely be closer to the mean while a larger one will be more spread out.

Sample use

```
import random
mean = 50

standard_dev = 2
print("The two numbers printed below should be pretty close to 50")
print(random.gauss(mean,standard_dev))
print(random.gauss(mean,standard_dev))

standard_dev2 = 100
print("The two numbers printed below should be pretty far from 50")
print(random.gauss(mean,standard_dev2))
print(random.gauss(mean,standard_dev2))
```

Your task

Write a program called ***random_tips.py*** which

- Takes as input a correct cooking time, cooking time standard deviation in the following form: “[Correct time],[standard deviation]”,
- Uses functions in the random module to determine actual cooking time and whether a further random tip is applied (see supporting information)
    - The actual cooking time should be determined using the `random.gauss()` function,
    - Whereas `random.random()` is used to decide whether a particular instance should result in a tip paid (this is based on the frequencies given in the supporting information),
- Prints the resultant cooking time and tip paid in the form shown below: “Actual cooking time was [random cooking time] and the tip paid was [final tip]%",
- Assume that **smaller numbers** result in **positive tips** and larger with negative.

Testing your work

In order to check whether your program is working correctly, you may like to print out the state of different variables with text explaining how this is being used (this will help you understand the code responds to the scenario correctly).

For instance, you could print:

the random value was: [uniform random result] which is in the range [category range] resulting in [final tip]”

```
Actual cooking time was 13016.476351274048 and the tip paid was 5%
The random value was 0.05304440911909081 which is in the range 0 to 0.1 matching a positive tip
```

You will need to decide for yourselves what will be meaningful for your own testing purposes

You will need to use

random.random

random.gauss

Sample input and output

Input 1

```
10000,5000
```

Output 1

```
Actual cooking time was 13016.476351274048 and the tip paid was 5%
```

Input 2

```
100,1
```

Output 2

```
Actual cooking time was 100.300872178288 and the tip paid was 0%
```

Input 3

```
10,10
```

Output 3

```
Actual cooking time was 8.560599784945515 and the tip paid was -5%
```

R.1,2,3,4,5 and 7. Restructuring earlier components into functions

Later tasks will have you having to repeat earlier parts multiple times which makes it a perfect time to convert your existing code into functions.

Your task

✕ Change 5 - Tuesday 29/03/2022 12:05pm Melbourne time:
We have updated the test for classify_cook_time in the file *test_restaurant_simulator.py* in the original workspace. You will need to grab the new file and replace your own (if don't already have the most recent one). Thanks!

The list below shows the programs from R1-5 and R7, corresponding function name, arguments and expected return table.

Prepare a file called ***restaurant_simulator.py*** which **contains** (at least) each of the functions defined below. Ensure that your function names match the ones we provide, as the automated marking will use these names for testing.

Note: The `display_menu()` function will use the print operation and the `get_meals_list_from_user()` will use the input operation but all others in this list should work exclusively using arguments and return values.

You will need to use

Your code from R1-5 and R7

Function list

R1. `get_meals_list_from_user()` details

▶ Expand

R2. `display_menu(options)` details

▶ Expand

R3. `validate_user_choice(options, users_input)` details

▶ Expand

R5. `classify_cook_time(average_cook_time, stdev_cook_time, actual_cook_time)` details

▶ Expand

R5. `get_cooking_tip(classification, base_tip)` details

▶ Expand

R7. `random_tip_compute(tip_chance, base_tip_value, random_comparison)` details

▶ Expand

R.8. Calculate profit on meal

For this task you will need to use your code from prior tasks (reorganised into functions as needed) to take an order and determine profit.

For the moment, you don’t need to worry about having to recook meals, we just want to take the order at the moment and return the profit on that meal.

How to test your program

This task onwards will not include automated tests due to their reliance on random values. However, if all your previous functions are correctly defined and have passed their tests, any errors arising should only occur within new code you produce.

We recommend setting up some variables to allow you to override the random values going into your functions so you can confirm how your program operates in the unique scenarios.

Be aware: the chance of a meal being slightly over/under cooked is less than one in three and the chance of being very under/over cooked is about a one in twenty chance. You should avoid trying to test these scenarios by running your code until you happen to get the scenario by chance as this is not a good use of your time.

Your task

Add a function to *restaurant_simulator.py* called `order(options)` as shown below.

Be sure to also add the following to the bottom of your program. (See e.g. here to understand what this does).

```
if __name__ == "__main__":
    #code to run your order function goes here
```

`Order(options)` details

Purpose:

- Displays the menu, allows user to choose a meal, determines cooking time and corresponding tip and the uniform random tip and computes the profit made on the order. All of this is done for the meal chosen by the user.
    - if you receive an invalid customer choice, the program should continue requesting input until a valid choice is given
- To help make your program clearer to a user, you should print
    - The meal chosen, how it was cooked (with actual and correct cooking time shown), the value of the two tips, the final selling price and the corresponding profit,

    - Floating point numbers rounded (or truncated) to 2 decimal places,
    - **Note:** you have flexibility with exactly how you display things, the point is to ensure the user knows what's going on.
- **Note:** To allow you to test this code, you will need to run (at least) `get_meals_list_from_user` prior to running the `order` method.

Arguments:

1. The variable `options`

Return Value:

- Profit made on this meal

Sample input and output:

- **Function call 1**

```
>>options = get_meals_list_from_user()
>>.
>>order(options)
```

- **Sample output**

```
Please enter your order. The options are given below
1. Name:Budda Bowl (vg) Sells:$25 Costs:$20 Takes:10.0 mins
2. Name:Eye Fillet Steak Sells:$55 Costs:$25 Takes:7.0 mins
3. Name:Spaghetti Bolognese Sells:$30 Costs:$22 Takes:40.0 mins
4. Name:Pad Thai (seafood) Sells:$22 Costs:$17 Takes:30.0 mins
please enter a number to make your choice
```

- **Input**

```
>>1
```

- **Output**

```
now cooking Budda Bowl (vg)
Budda Bowl (vg) was well cooked (11.04 vs 10 mins)
cooking tip was 10 random tip was 0 the random value being (0.83)
final selling price was $27.50
for a profit of $7.50
```

- **Return value 1**

```
7.5
```

- **Function call 2**

```
>>options = get_meals_list_from_user()
```

```
>>.
>>order(options)
```

- **Sample output**

```
Please enter your order. The options are given below
1. Name:Budda Bowl (vg) Sells:$25 Costs:$20 Takes:10 mins
2. Name:Eye Fillet Steak Sells:$55 Costs:$25 Takes:7 mins
3. Name:Spaghetti Bolognese Sells:$30 Costs:$22 Takes:40 mins
4. Name:Pad Thai (seafood) Sells:$22 Costs:$17 Takes:30 mins
please enter a number to make your choice
```

- **Input**

```
>>2
```

- **Output**

```
now cooking Eye Fillet Steak
Eye Fillet Steak was slightly overcooked (8.4 vs 7 mins)
cooking tip was 0 random tip was 5 the random value being (0.08)
final selling price was $57.75
for a profit of $32.75
```

- **Return value 2**

```
32.75
```

- **Function call 3**

```
>>options = get_meals_list_from_user()
>>.
>>order(options)
```

- **Sample output**

```
Please enter your order. The options are given below
1. Name:Budda Bowl (vg) Sells:$25 Costs:$20 Takes:10 mins
2. Name:Eye Fillet Steak Sells:$55 Costs:$25 Takes:7 mins
3. Name:Spaghetti Bolognese Sells:$30 Costs:$22 Takes:40 mins
4. Name:Pad Thai (seafood) Sells:$22 Costs:$17 Takes:30 mins
please enter a number to make your choice
```

- **Input**

```
>>1
```

- **Output**

```
now cooking Budda Bowl (vg)
```

```
Budda Bowl (vg) was very undercooked (3.7 vs 10 mins)
cooking tip was -100 random tip was -5 the random value being (1)
final selling price was $0.00
for a profit of -$20.00
```

Note: while we have displayed the negative profit as `-$20.00` you could equally write `$-20.00` it's just about being clear what's going on.

- **Return value 3**

```
-20.0
```

You will need to use

The functions you have previously prepared in this assignment

R.6. Retry cooking a meal up to twice more if very over or undercooked

Your task

Update your `order` function in *restaurant_simulator.py* so that it:

- Checks whether a meal is very overcooked or very undercooked,
- If this is the case, it will simulate “recooking” the **same** meal by generating a new cooking time using the `random.gauss` function again,
- This “recooking” should occur up to 3 times before giving up,
- Each failed attempt at cooking the meal should add to the costs for the restaurant.

How to test your program

Once again, automated tests are not provided and you should rely on your ability to override the random values in particular ways to test the different scenarios.

You will need to use

The functions you have previously prepared in this assignment

Sample input and output

Below shows running the order method with cooking times overridden to be very overcooked

Function call

```
>>options = get_meals_list_from_user()
>>.
>>order(options)
>>1
```

Sample output

```
now cooking Budda Bowl (vg)
Budda Bowl (vg) was very overcooked (16.3 vs 10 mins)
cooking tip was -100 random tip was 0 the random value being (0.83)
final selling price was $0.00
for a profit of -$20.00
Budda Bowl (vg) was very overcooked (16.3 vs 10 mins)
cooking tip was -100 random tip was 0 the random value being (0.89)
final selling price was $0.00
for a profit of -$20.00
Budda Bowl (vg) was very overcooked (16.3 vs 10 mins)
```

```
cooking tip was -100 random tip was -5 the random value being (0.96)
final selling price was $0.00
for a profit of -$20.00
giving up after 3 failed attempts
overall, the profit for this meal was -$60.00
```

Return value

```
-60.0
```

Function call 2

```
>>options = get_meals_list_from_user()
>>.
>>order(options)
>>1
```

Sample output 2

```
now cooking Budda Bowl (vg)
Budda Bowl (vg) was very overcooked (16.3 vs 10 mins)
cooking tip was -100 random tip was 5 the random value being (0)
final selling price was $0.00
for a profit of -$20.00
Budda Bowl (vg) was slightly undercooked (6.7 vs 10 mins)
cooking tip was 0 random tip was 0 the random value being (0.52)
final selling price was $25.00
for a profit of $5.00
overall, the profit for this meal was -$15.00
```

Return value

```
-15.0
```

R.9. Repeat steps 1-8 for each customer in the restaurant and determine overall profit

Your task

Add a function to *restaurant_simulator.py* called `order_for_x_people(X)` as per the below.

Remember to use the `if __name__ == "__main__"` block to run your program (5 customers should be sufficient).

`order_for_x_people(X)` details

▶ Expand

You will need to use

The functions you have previously prepared in this assignment

Do not forget to submit on Moodle

See this slide.

*** END OF FIT1045 ASSIGNMENT 2***

FIT1053 Only Component

In this assignment, we have run on the assumption for R4 that the **average** / mean cooking time for a meal **was also** the **perfect** cooking time for that meal; however this is not strictly true as the average cooking time (and variability around that) depends on those **doing** the cooking.

Write a program **restaurant_simulator_sampling.py** which allows for meals where the perfect and average cooking time are potentially **different**.

Create a menu with just one (1) meal item and look at what happens over a large number of simulations for the **same perfect** cooking time but **different average** cooking times and standard deviations.

Then respond to the following questions and include the data from your simulations you are using to justify your argument:

> **What impact would a more experienced vs less experienced chef have on the profitability of a restaurant?**

As part of your response be sure to include:

- Your assumptions about what a more experience chef looks like in terms of average and standard deviation of cooking times,
- At least 10 unique combinations of average and standard deviation,
- A data set involving repeated measurements of the same combinations to ensure reliability of your data.

You may find it helpful to update this program so that...

- A sequence of meal selections to be made at once (e.g. 1,1,1,1,1 to represent cooking meal option 1 five times in a row),
- Output can be suppressed,
- The total profit for a sequence of customers to be written to a list or other structure.

Do not forget to submit on Moodle

See this slide.

END OF FIT1053 ASSIGNMENT 2