Exemplar_ Conditional statements

November 2, 2023

Exemplar: Conditional Statements

0.1 Introduction

In this lab, you will practice using Python operators to perform operations between two variables and write conditional statements.

As a data analyst, you'll use conditional statements in your approach to many different tasks using Python. Using Boolean values, conditional statements will determine how a code block is executed based on how a condition is met.

While continuing work for the movie theater, you're now examining marketing campaigns. Specifically, you'll be analyzing user behavior to determine when a customer's activity has prompted a marketing email.

0.2 Tips for completing this lab

As you navigate this lab, keep the following tips in mind:

- `### YOUR CODE HERE ###` indicates where you should write code. Be sure to replace this with your own code before running the code cell.
- Feel free to open the hints for additional guidance as you work on each task.
- To enter your answer to a question, double-click the markdown cell to edit. Be sure to replace the "[Double-click to enter your responses here.]" with your own answer.
- You can save your work manually by clicking File and then Save in the menu bar at the top of the notebook.
- You can download your work locally by clicking File and then Download and then specifying your preferred file format in the menu bar at the top of the notebook.

0.3 Task 1: Define a comparator function

You are a data professional for a movie theater, and your task is to use customers' purchasing data to determine whether or not to send them a marketing email.

- Define a function called `send_email` that accepts the following arguments:
  - `num_visits` - the number of times a customer has visited the theater
  - `visits_email` - the minimum number of visits the customer must have made for them to receive a marketing email

- The function must print either: `Send email.` or `Not enough visits.`

Example:

```
 [IN] send_email(num_visits=3, visits_email=5)
[OUT] 'Not enough visits.'

 [IN] send_email(num_visits=5, visits_email=5)
[OUT] 'Send email.'
```

Note that there is more than one way to solve this problem.

[1]:
```
### YOUR CODE HERE ###
def send_email(num_visits, visits_email):
    if num_visits >= visits_email:
        print('Send email.')
    else:
        print('Not enough visits.')
```

Hint 1

Consider the syntax for defining a function. Remember: a function definition statement must include a `def` keyword, the function's name, and its arguments, followed by a colon.

Hint 2

Recall Python's conditional and comparison operators. Check your indentation. Is your code indented properly?

Hint 3

One approach is to compare `num_visits` to `visits_email` using the `>=` comparator. If `num_visits >= visits_email`, print `Send email.` Otherwise, print `Not enough visits.`.

0.3.1 Test your function

Test your function against the following cases by running the cell below.

[2]:
```
send_email(num_visits=3, visits_email=5)    # Should print 'Not enough visits.'
send_email(num_visits=5, visits_email=5)    # Should print 'Send email.'
send_email(num_visits=15, visits_email=10)  # Should print 'Send email.'
```

```
Not enough visits.
Send email.
Send email.
```

0.4 Task 2: Add logical branching to your function

The theater is offering a promotion where customers who have visited the theater more than a designated number of times will also receive a coupon with their email. Update the function that you created above to include additional logical branching.

- Include an additional argument `visits_coupon` that represents the minimum number of visits the customer must have made for them to receive a coupon with their email.
- The function must print one of three possible messages:
    1. `Send email with coupon.`
    2. `Send email only.`
    3. `Not enough visits.`

The minimum number of visits to receive a coupon will always be greater than the number to receive an email only.

Example:

```
 [IN] send_email(num_visits=3, visits_email=5, visits_coupon=8)
[OUT] 'Not enough visits.'

 [IN] send_email(num_visits=5, visits_email=5, visits_coupon=8)
[OUT] `Send email only.`

 [IN] send_email(num_visits=8, visits_email=5, visits_coupon=8)
[OUT] `Send email with coupon.`
```

Note that there is more than one way to solve this problem.

[3]:
```
### YOUR CODE HERE ###
def send_email(num_visits, visits_email, visits_coupon):
    if num_visits >= visits_coupon:
        print('Send email with coupon.')
    elif num_visits >= visits_email:
        print('Send email only.')
    else:
        print('Not enough visits.')
```

Hint 1

Refer to what you've learned about conditional statements, logical operators, and comparison operators.

Hint 2

Make sure your `if`, `elif`, and `else` statements are indented properly beneath the function's definition line.

Make sure your print statements are indented properly beneath each conditional statement.

Check syntax.

Hint 3

One approach is to compare `num_visits` to both `visits_coupon` and `visits_email` using the `>=` comparator:

If `num_visits >= visits_coupon`, print `Send email with coupon.` Or else if `num_visits >= visits_email`, print `Send email only.` Otherwise, print `Not enough visits.`

0.4.1 Test your function

Test your function against the following cases by running the cell below.

[4]:
```
send_email(num_visits=3, visits_email=5, visits_coupon=8)   # Should print 'Not␣
↪enough visits.'
send_email(num_visits=5, visits_email=5, visits_coupon=8)   # Should print␣
↪'Send email only.'
send_email(num_visits=6, visits_email=5, visits_coupon=8)   # Should print␣
↪'Send email only.'
send_email(num_visits=8, visits_email=5, visits_coupon=8)   # Should print␣
↪'Send email with coupon.'
send_email(num_visits=10, visits_email=5, visits_coupon=8)  # Should print␣
↪'Send email with coupon.'
```

```
Not enough visits.
Send email only.
Send email only.
Send email with coupon.
Send email with coupon.
```

0.5 Conclusion

What are your key takeaways from this lab? * Conditional statements, comparison operators, and logical operators play a major role in automating processes and controlling the branching of code execution. * Conditional statements allow you to determine whether a specific set of criteria has been met. * Comparison operators allow you to compare pairs of values. * Comparison operators such as `>=` and `<=` can be combined to select a range of data within a specific parameter. * Logical operators such as `and` and `or` allow you to check more than one condition at a time.

Congratulations! You've completed this lab. However, you may not notice a green check mark next to this item on Coursera's platform. Please continue your progress regardless of the check mark. Just click on the "save" icon at the top of this notebook to ensure your work has been logged.