INTERVIEW EXERCISES

CS 3030: Python

Instructor: Damià Fuentes Escoté

UCCS University of Colorado Colorado Springs

Next class – Bring laptop

- Bring the laptop as we are going to do examples of interview exercises.

Last quiz – Quiz 6 – 7/5/19 – Class before the final term

Preparing for the coding interview

- Every company has a different way of interviewing its candidates, and you need to be prepared for all those possible scenarios.
- The smaller the company, the less sophisticated the interviewing structure. Smaller companies and startups will probably only do 1 - 2 live coding interviews (interactive interviews with a tool like Codepen or an actual IDE, or whiteboard interviews) or they might give you a small project to complete on your own.

https://medium.freecodecamp.org/the-ultimate-guide-to-preparing-for-the-coding-interview-183251ee36c9

INTERVIEW EXERCISES

Preparing for the coding interview

- Large companies such as Google or Facebook may have a lot different stages:
  - *Recruiter phone interview – explain me what you did in this project…*
  - *Technical phone interview – what is your favorite search method?*
  - *Onsite interview*
  - *Whiteboard coding interview*
  - *Coding interview – solve the palindrome problem*
  - *Design interview – ability to work with complex and scalable services*
  - *Cultural fit interview – company culture fit*
  - *Etc*

Preparing for the coding interview

- Large companies such as Google or Facebook may have a lot different stages:
  - *Recruiter phone interview*
  - *Technical phone interview*
  - *Onsite interview*
  - *Whiteboard coding interview*
  - *Coding interview*
  - *Design interview*
  - *Cultural fit interview*
  - *Etc*

All of them have coding interviews! Online and then Onsite. Prerequisite for getting an offer.

Coding interviews done online are meant to filter people. They may not even look to your code. If your code passes some tests, then you pass to the next stage.

If you cannot code/solve the problem (or have some serious bugs in your code), it's quite difficult to get an offer.

Interview formats

- If you are curious here is a link on how different companies structure their different interviews.
  - *https://github.com/yangshun/tech-interview-handbook/blob/master/non-technical/interview-formats.md*

Resources to prepare for the coding interview

- Hackerrank
  - *https://www.hackerrank.com*
- Learneroo
  - *https://www.learneroo.com*
- Educative
  - *https://www.educative.io*
- Leetcode
  - *https://leetcode.com*
- Coderbyte
  - *https://coderbyte.com*
- Cracking the Coding Interview: 189 Programming Questions and Solutions
  - *https://www.amazon.com/Cracking-Coding-Interview-Programming-Questions/dp/0984782850/*

STEPS TO SOLVING A PROGRAMMING PROBLEM

STEPS TO SOLVING A PROGRAMMING PROBLEM

Not a coding interview problem!
But it will help.

Steps to solving a programming problem

- 10 steps to solving a programming problems by Valinda Chan
  - *https://codeburst.io/10-steps-to-solving-a-programming-problem-8a32d1e96d74*
  - *You have to understand them. Later, make your own steps, that will work best for you.* ***Everybody's mind is different!***
  - *The following steps is just some good advice.*
  - ***Programming problem != coding interview****, but you may apply most of the steps.*

Steps to solving a programming problem

- We are going to go through the steps by solving this problem:

  - Create a simple function *selectEvenNumbers* that will take in an array of numbers and return an array *evenNumbers* of only even numbers. If there are no even numbers, return the empty array *evenNumbers.*

Steps to solving a programming problem

- **1. Read the problem at least three times** (or however many makes you feel comfortable)
  - *You can't solve a problem you don't understand*
  - *There is a difference between the problem and the problem you think you are solving.*
  - *It's easy to start reading the first few lines in a problem and assume the rest of it because it's similar to something you've seen in the past. But there may be a trick!*
  - *You don't want to find out halfway through that you misunderstood the problem.*
  - ***The better you understand the problem, the easier it will be to solve it.***

https://codeburst.io/10-steps-to-solving-a-programming-problem-8a32d1e96d74

Steps to solving a programming problem

- Create a simple function *selectEvenNumbers* that will take in an array of numbers and return an array *evenNumbers* of only even numbers. If there are no even numbers, return the empty array *evenNumbers.*

- Create a simple function *selectEvenNumbers* that will take in an array of numbers and return an array *evenNumbers* of only even numbers. If there are no even numbers, return the empty array *evenNumbers.*

- Create a simple function *selectEvenNumbers* that will take in an array of numbers and return an array *evenNumbers* of only even numbers. If there are no even numbers, return the empty array *evenNumbers.*

Steps to solving a programming problem

- **How can a computer tell what is an even number?**
  - *Divide that number by 2 and see if its remainder is 0.*
- **What am I passing into this function?**
  - *An array*
- **What will that array contain?**
  - *One or more numbers, or maybe is empty*
- **What are the data types of the elements in the array?**
  - *Numbers*
- **What is the goal of this function? What am I returning at the end of this function?**
  - *The goal is to take all the even numbers and return them in an array. If there are no even numbers, return an empty array.*

Steps to solving a programming problem

- **2. Work through the problem manually with at least three sets of sample data**
  - *Take out a piece of paper and work through the problem manually. Think of at least three sets of sample data you can use. Consider corner and edge cases as well:*
    - **Corner case**: a problem or situation that occurs outside of normal operating parameters, specifically when multiple environmental variables or conditions are simultaneously at extreme levels, even though each parameter is within the specified range for that parameter.
      - *Input range: 0-100, Normal inputs: 40-60, Corner cases: 90, 11, 3, 87, etc*
    - **Edge case**: problem or situation that occurs only at an extreme (maximum or minimum) operating parameter.
      - *Input range: 0-100, Normal inputs: 40-60, Edge cases: 0, 100*

Steps to solving a programming problem

- Some examples to use:

```
[1]

[1, 2]

[1, 2, 3, 4, 5, 6]

[-200.25]

[-800.1, 2000, 3.1, -1000.25, 42, 600]
```

- Try using large sets of data as it will override your brain's ability to naturally solve the problem just by looking at it. That helps you work through the real algorithm.

Steps to solving a programming problem

- Let's go through the first array `[1]`
  - *Look at the only element in the array* `[1]`
  - *Decide if it is even. It is not*
  - *Notice that there are no more elements in this array*
  - *Determine there are no even numbers in this provided array*
  - *Return an empty array*

Steps to solving a programming problem

- Let's go through the array `[1, 2]`
  - *Look at the first element in array* `[1, 2]`
  - *It is 1*
  - *Decide if it is even. It is not*
  - *Look at the next element in the array*
  - *It is 2*
  - *Decide if it is even. It is even*
  - *Make an array evenNumbers and add 2 to this array*
  - *Notice that there are no more elements in this array*
  - *Return the array evenNumbers which is* [2]

Steps to solving a programming problem

- **3. Simplify and optimize your steps**
  - *Look for patterns and see if there's anything you can generalize. See if you can reduce any steps or if you are repeating any steps.*

Steps to solving a programming problem

1. Create a function selectEvenNumbers
2. Create a new empty array evenNumbers where I store even numbers, if any
3. Go through each element in the array [1, 2]
4. Find the first element
5. Decide if it is even by seeing if it is divisible by 2. If it is even, I add that to evenNumbers
6. Find the next element
7. Repeat step #5
8. Repeat step #6 and #5 until there are no more elements in this array
9. Return the array evenNumbers, regardless of whether it has anything in it

Steps to solving a programming problem

- **4. Write pseudocode**
  - *Even after you've worked out general steps, writing out pseudocode that you can translate into code will help with defining the structure of your code and make coding a lot easier.* ***Write pseudocode line by line.***
  - ***Don't get caught up with the syntax. Focus on the logic and steps.***

Steps to solving a programming problem

- Option 1 (more words):

```
function selectEvenNumbers

    create an array evenNumbers and set that equal to an empty array

    for each element in that array
        see if that element is even
            if element is even (if there is a remainder when divided by 2)
                add that to the array evenNumbers

    return evenNumbers
```

Steps to solving a programming problem

- Option 2 (less words):

```
function selectEvenNumbers

    evenNumbers = []

    for i = 0 to i = length of evenNumbers
        if (element % 2 === 0)
            add that to the array evenNumbers

    return evenNumbers
```

Steps to solving a programming problem

- Option 2 (less words):

Either way is fine as long as you are writing it out line-by-line and understand the logic on each line.

```
function selectEvenNumbers

    evenNumbers = []

    for i = 0 to i = length of evenNumbers
        if (element % 2 === 0)
            add to that to the array evenNumbers

    return evenNumbers
```

Steps to solving a programming problem

- **5. Translate pseudocode into code, debug and test**
  - *When you have your pseudocode ready, translate each line into real code in the language you are working on.*
    - Python? Java?
  - *Call the function and give it some sample sets of data you used earlier. Use them to see if your code returns the results you want.*
  - *Generally use `logs` or `prints` after each variable or line or so. This helps check if the values and code are behaving as expected before you move on. By doing this, you can catch any issues before you get too far.*

Steps to solving a programming problem

```
def selectEvenNumbers(numbers):

    evenNumbers = []

    for num in numbers:
        if num % 2 == 0:
            evenNumbers.append(num)

    return evenNumbers
```

Steps to solving a programming problem

```
print(selectEvenNumbers([1]))
print(selectEvenNumbers([1, 2]))
print(selectEvenNumbers([1, 2, 3, 4, 5, 6]))
print(selectEvenNumbers([-200.25]))
print(selectEvenNumbers([-800.1, 2000, 3.1, -1000.25, 42, 600]))
# []
# [2]
# [2, 4, 6]
# []
# [2000, 42, 600]
```

Steps to solving a programming problem

- **6. Simplify and optimize your code**

“Simplicity is prerequisite for reliability.” – Edsger W. Dijkstra

  - Simplifying and optimizing your code may require you to iterate a few times, identifying ways to further simplify and optimize code.

Steps to solving a programming problem

```
def selectEvenNumbers(numbers):

    evenNumbers = []

    for num in numbers:
        if num % 2 == 0:
            evenNumbers.append(num)

    return evenNumbers
```

Steps to solving a programming problem

```
def selectEvenNumbers(numbers):
    return [num for num in numbers if num % 2 == 0]
```

Steps to solving a programming problem

- **What are your goals for simplifying and optimizing?**
  - *The goals will depend on your team's style or your personal preference. Are you trying to condense the code as much as possible? Is the goal to make it the code more readable? If that's the case, you may prefer taking that extra line to define the variable or compute something rather than trying to define and compute all in one line.*
- **Are there any more extra steps you can take out?**
- **Are there any variables or functions you ended up not even needing or using?**
- **Are you repeating some steps?**
  - *See if you can define in another function.*
- **Are there better ways to handle edge cases?**

Steps to solving a programming problem

- **7. Debug**
  - *This step really should be throughout the process.*
  - *Debugging throughout will help you catch any syntax errors or gaps in logic sooner rather than later.*
  - *Take advantage of your Integrated Development Environment (IDE) and debugger. When you encounter bugs, you should trace the code line-by-line to see if there was anything that did not go as expected.*
  - *Comment out chunks or lines of code and output what you have so far to quickly see if the code is behaving how you expected. You can always uncomment the code as needed.*
  - *Use other sample data if there are scenarios you did not think of and see if the code will still work.*
  - *Save different versions of your file if you are trying out a completely different approach. You don't want to lose any of you work if you end up wanting to revert back to it!*

Steps to solving a programming problem

- **8. Write useful comments**
  - *You may not always remember what every single line meant a month later. And someone else working on your code may not know either. That's why it's important to write useful comments to avoid problems and save time later on if you need to come back to it.*
  - ***Stay away from comments such as**:*

    ```
    # This is an array. Iterate through it.
    # This is a variable
    ```
  - *Try to write brief, high-level comments that help you understand what's going on if it is not obvious.*
    - What is this code for?
    - What is it doing?

Steps to solving a programming problem

- **9. Get feedback through code reviews**
  - *Get feedback from your teammates, professors, and other developers. Check out Stack Overflow. See how others tackled the problem and learn from them. There are sometimes several ways to approach a problem. Find out what they are and you'll get better and quicker at coming up with them yourself.*

"No matter how slow you are writing clean code, you will always be slower if you make a mess." – Uncle Bob Martin

Steps to solving a programming problem

- **10. Practice, practice, practice**
  - *Even experienced developers are always practicing and learning. If you get helpful feedback, implement it. Redo a problem or do similar problems. Keep pushing yourself.* ***With each problem you solve, the better a developer you become.*** *Celebrate each success and be sure to remember how far you've come. Remember that programming, like with anything, comes easier and more naturally with time.*

"Take pride in how far you've come. Have faith in how far you can go. But don't forget to enjoy the journey." – Michael Josephson

HOW LARGE COMPANIES FILTER PEOPLE IN THE FIRST STAGES

How large companies filter people in the first stages

- Two files:
  - *One file with a function you have to fill*
  - *Another file that you don't see where they test the implemented function with a lot of test cases.*

- Let's develop that for the midterm palindrome problem.

Palindrome problem

Problem: Write a Python function that checks whether a passed string is palindrome or not.

Note: A palindrome is a word, phrase, or sequence that reads the same backward as forward.

Example 1 Input: “madam”

Output: True

Example 2 Input: “Hello world!”

Output: False

Example 3 Input: “Eva, Can I Stab Bats In A Cave?”

Output: True

File or editor given to you

```
def is_palindrome(str):
    # Write your code here
    return True
```

File they will execute once you submit (You don’t see it)

Next slide

```python
from .run import is_palindrome
import unittest

class TestPalindrome(unittest.TestCase):
    def test_1(self):
        self.assertEqual(is_palindrome('madam'), True, "Should be True")

    def test_2(self):
        self.assertEqual(is_palindrome('redder'), True, "Should be True")

    def test_3(self):
        self.assertEqual(is_palindrome('Eva, Can I Stab Bats In A Cave?'), True, "Should be True").

    # more tests...

    def test_10(self):
        with self.assertRaises(AssertionError):
            is_palindrome(9)

if __name__ == '__main__’:
    unittest.main()
```

```
def is_palindrome(var):
    return var == var[::-1]
```

Output:

```
Ran 10 tests in 0.011s

FAILED (failures=5, errors=1)
```

```
def is_palindrome(var):
    var = var.lower()
    return var == var[::-1]
```

Output:

```
Ran 10 tests in 0.011s
FAILED (failures=4, errors=1)
```

```
def is_palindrome(var):
    namesRegex = re.compile(r'''[^a-zA-Z]''')
    var = namesRegex.sub('', var)
    var = var.lower()
    return var == var[::-1]
```

Output:

```
Ran 10 tests in 0.011s

FAILED (errors=1)
```

```
def is_palindrome(var):
    assert isinstance(var, str), 'The argument should be a string'
    namesRegex = re.compile(r'''[^a-zA-Z]'''')
    var = namesRegex.sub('', var)
    var = var.lower()
    return var == var[::-1]
```

Output:

```
Ran 10 tests in 0.011s

OK
```

```
"""Problem: Write a function to find if two strings are rotations
of each other.

Example 1   Input: "ABC", "BCA"
            Output: True

Example 2    Input: "ABC", "CAB"
            Output: True

Example 3    Input: "ABC", "CBA"
            Output: False
"""

def string_rotation(var1, var2):
    # Your code here
    pass
```

```
"""Problem:  Find all pairs in an array of integers whose sum is
equal to the given number.

Example 1   Input: [1, 2, 3], 5
            Output: [(2, 3), (3, 2)]

Example 2    Input: [1, 2, 3, 4, 5], 8
            Output: [(3, 5), (4, 4), (5, 3)]

Example 3    Input: [1, 2, 3], 8
            Output: []
"""

def find_pairs(array, num):
    # Your code here
    pass
```