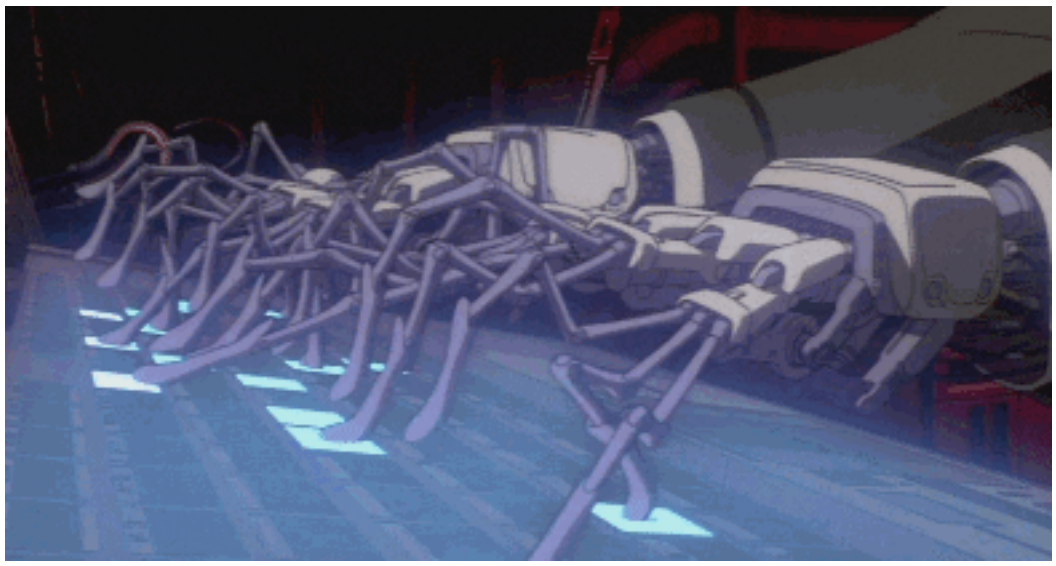
Workshop



A: Up to date

Go through the lecture and:

1. Make sure that you understand
    - Algorithmic complexity and Big-O notation
    - Abstract data types
        - Stacks and Queues
2. Implement the remainder of your `Stack` implementation
    - `pop`, `is_empty`, `size`
3. Look at the slides for the `Queue` abstract data type. Create the `Queue` class in Mu, as shown in the slide, and implement the missing code for the `is_empty` and `size` methods

B: Add more Checks to a Function

After the lecture, the function `balanced_quotation_marks` does not work properly as intended.

```
def balanced_quotation_marks(text):
    check_stack = Stack()

    for character in text:
        if character == '“':
            check_stack.push(character)
        elif character == '”':
            check_stack.pop()

    if check_stack.is_empty():
        print('I think you quotation signs are balanced.')
        return True
    else:
        print('Hov, it seems as if you forgot to unquote text.')
        return False
```

For example, with the following call to the function `balanced_quotation_marks` makes it behave not as intended:

```
balanced_quotation_marks('As to the latter part, I have no means of check
ing you," said I, "but"')
```

- Try to understand what is going wrong. If in doubt run the program in the debugger and *click* through it to understand what is going on.
- Modify the function `balanced_quotation_marks` so that it produces a correct error message when an error occurs as above.
- **Hint** you may want to recover from a crashing program. If in doubt about how to do that, check session 4.

C: Add Checks for Other Characters

There are other quotes such as `‘` and `’` and additionally there are parenthesis `(` and `)` all of which should be balanced in a text. See for example the following excerpt from *The Hounds of Baskervilles*.

In [2]:

```
def get_text_from_file(path_to_file):
    with open(path_to_file) as fp:
        content_lines = fp.readlines()
    return content_lines


lines = get_text_from_file('the_hound_of_the_baskervilles.txt')
text_for_analysis = ''.join(lines[146:172])
print(text_for_analysis)
```

```
“As to the latter part, I have no means of checking you,” said I, “b
ut
at least it is not difficult to find out a few particulars about the
man’s age and professional career.” From my small medical shelf I to
ok
down the Medical Directory and turned up the name. There were severa
l
Mortimers, but only one who could be our visitor. I read his record
aloud.

        “Mortimer, James, M.R.C.S., 1882, Grimpen, Dartmoor, Devon.
        House-surgeon, from 1882 to 1884, at Charing Cross Hospital.
        Winner of the Jackson prize for Comparative Pathology,
        with essay entitled ‘Is Disease a Reversion?’  Corresponding
        member of the Swedish Pathological Society.  Author of
        ‘Some Freaks of Atavism’ (Lancet 1882).  ‘Do We Progress?’
        (Journal of Psychology, March, 1883).  Medical Officer
        for the parishes of Grimpen, Thorsley, and High Barrow.”

“No mention of that local hunt, Watson,” said Holmes with a mischiev
ous
smile, “but a country doctor, as you very astutely observed. I think
that I am fairly justified in my inferences. As to the adjectives, I
said, if I remember right, amiable, unambitious, and absent-minded.
It is my experience that it is only an amiable man in this world who
receives testimonials, only an unambitious one who abandons a London
career for the country, and only an absent-minded one who leaves his
stick and not his visiting-card after waiting an hour in your room.”

“And the dog?”
```

Modify the function `balanced_quotation_marks` so that it consumes three arguments, where the first one is the `text` to check, the second is an *opening* character, such as `“`, `‘`, or `(`, and the third is a *closing* character, such as `”`, `’`, or `)`.

```
def balanced_quotation_marks(text, opening='“', closing='”'):
    # TODO: Implement me
    pass
```

Consequently, you can check if various characters are correclty balanced.

D: Program Which Checks Various Characters

Write a function `check_all`, which gets next to a text, a list of pairs of characters and sequentially checks if they are all balanced in a text. Let the body of the function `check_all` contain a `for`-loop over the `list_of_characters` to get the corresponding pairs and reuse the function `balanced_quotation_marks` within that `for`-loop.

```
def check_all(text, list_of_characters):
    # TODO: Implement me!
    pass


characters = [('“', '”'), ('(', ')'), ('‘', '’')]
check_all(text_for_analysis, characters)
```

Hints:

- Reuse the function `balanced_quotation_marks(text, opening='“', closing='”')` from above.
- Remember how to destructure lists, tuples, etc.

In [ ]:

```
lines = get_text_from_file('the_hound_of_the_baskervilles.txt')
text_for_analysis = ''.join(lines[146:172])
characters = [('“', '”'), ('(', ')'), ('‘', '’')]

check_all(text_for_analysis, characters)
```

E: What is this code doing...?

Copy and paste the following program into a file called `mysterious.py`.

After typing it in, examine it's behavior by step-wise execution in the debugger.

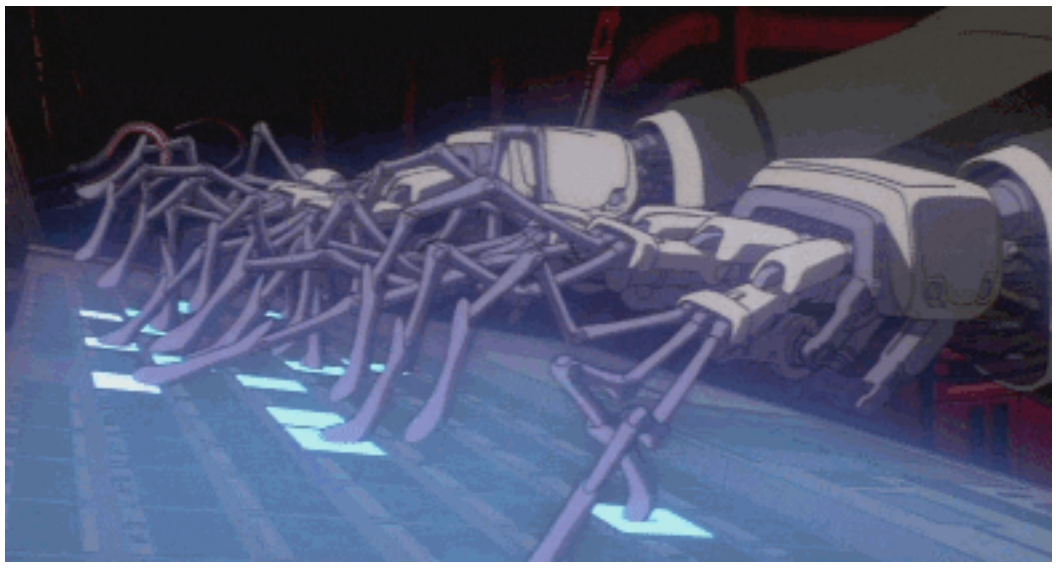
- Hint: set a breakpoint on line 2.

1. Describe what this program is actually doing.
2. What is the running time complexity of the algorithm. Is it:
   A. Logarithmic
   B. Linear
   C. Quadratic
   D. Cubic

```
def mysterious_function(data_list):
    for passnum in range(len(data_list) - 1, 0, -1):
        for idx in range(passnum):
            if data_list[idx] > data_list[idx + 1]:
                temp = data_list[idx]
                data_list[idx] = data_list[idx + 1]
                data_list[idx + 1] = temp

if __name__ == '__main__':
    data = [54, 26, 93, 17, 77, 31, 44, 55, 20]
    mysterious_function(data)
    print(data)
```

Workshop



A: Up to date

Go through the lecture and:

1. Make sure that you understand
   - Algorithmic complexity and Big-O notation
   - Abstract data types
     - Stacks and Queues
2. Implement the remainder of your `Stack` implementation

   - `pop`, `is_empty`, `size`
3. Look at the slides for the `Queue` abstract data type. Create the `Queue` class in Mu, as shown in the slide, and implement the missing code for the `is_empty` and `size` methods

B: Add more Checks to a Function

After the lecture, the function `balanced_quotation_marks` does not work properly as intended.

```
def balanced_quotation_marks(text):
    check_stack = Stack()

    for character in text:
        if character == '“':
            check_stack.push(character)
        elif character == '”':
            check_stack.pop()

    if check_stack.is_empty():
        print('I think you quotation signs are balanced.')
        return True
    else:
        print('Hov, it seems as if you forgot to unquote text.')
        return False
```

For example, with the following call to the function `balanced_quotation_marks` makes it behave not as intended:

```
balanced_quotation_marks('As to the latter part, I have no means of check
ing you,” said I, “but”')
```

- Try to understand what is going wrong. If in doubt run the program in the debugger and *click* through it to understand what is going on.
- Modify the function `balanced_quotation_marks` so that it produces a correct error message when an error occurs as above.
- **Hint** you may want to recover from a crashing program. If in doubt about how to do that, check session 4.

C: Add Checks for Other Characters

There are other quotes such as ‘ and ’ and additionally there are parenthesis (and) all of which should be balanced in a text. See for example the following excerpt from *The Hounds of Baskervilles*.

In [2]:

```
“As to the latter part, I have no means of checking you,” said I, “b
ut
at least it is not difficult to find out a few particulars about the
man’s age and professional career.” From my small medical shelf I to
ok
down the Medical Directory and turned up the name. There were severa
l
Mortimers, but only one who could be our visitor. I read his record
aloud.

      “Mortimer, James, M.R.C.S., 1882, Grimpen, Dartmoor, Devon.
      House-surgeon, from 1882 to 1884, at Charing Cross Hospital.
      Winner of the Jackson prize for Comparative Pathology,
      with essay entitled ‘Is Disease a Reversion?’  Corresponding
      member of the Swedish Pathological Society.  Author of
      ‘Some Freaks of Atavism’ (Lancet 1882).  ‘Do We Progress?’
      (Journal of Psychology, March, 1883).  Medical Officer
      for the parishes of Grimpen, Thorsley, and High Barrow.”
```

Modify the function `balanced_quotation_marks` so that it consumes three arguments, where the first one is the `text` to check, the second is an *opening* character, such as `“`, `‘`, or `(`, and the third is a *closing* character, such as `”`, `’`, or `)`.

```
def balanced_quotation_marks(text, opening='“', closing='”'):
    # TODO: Implement me
    pass
```

Consequently, you can check if various characters are correclty balanced.

D: Program Which Checks Various Characters

Write a function `check_all`, which gets next to a text, a list of pairs of characters and sequentially checks if they are all balanced in a text. Let the body of the function `check_all` contain a `for`-loop over the `list_of_characters` to get the corresponding pairs and reuse the function `balanced_quotation_marks` within that `for`-loop.

```
def check_all(text, list_of_characters):
    # TODO: Implement me!
    pass


characters = [('“', '”'), ('(', ')'), ('‘', '’')]
check_all(text_for_analysis, characters)
```

Hints:

- Reuse the function `balanced_quotation_marks(text, opening='“', closing='”')` from above.
- Remember how to destructure lists, tuples, etc.

In [ ]:

E: What is this code doing...?

Copy and paste the following program into a file called `mysterious.py`.

After typing it in, examine it's behavior by step-wise execution in the debugger.

- Hint: set a breakpoint on line 2.

1. Describe what this program is actually doing.
2. What is the running time complexity of the algorithm. Is it:
   A. Logarithmic
   B. Linear
   C. Quadratic
   D. Cubic

```
def mysterious_function(data_list):
    for passnum in range(len(data_list) - 1, 0, -1):
        for idx in range(passnum):
            if data_list[idx] > data_list[idx + 1]:
                temp = data_list[idx]
                data_list[idx] = data_list[idx + 1]
                data_list[idx + 1] = temp

if __name__ == '__main__':
    data = [54, 26, 93, 17, 77, 31, 44, 55, 20]
    mysterious_function(data)
    print(data)
```