ENGG1811: Computing For Engineers

2025 Term 2

Selection Structure

Week 3: Friday 20th June, 2025

Friday 16:00 - 18:00 | OboeME304

Today

Boolean Expressions

If/Elif/Else Statements

Plotting

Lab Tips

Reminders

- Assignment 1 will be released soon
- Help sessions will start next week. You can get one-on-one tutoring for content, labs, or assignments here.
- Live coding sessions are on Fridays from 1–2 PM, located Morven Brown G3
- PASS session still going on!
- Remember to ask for help if you need it!
- Make sure you check your marks in grades on the course website
- Logins to the course website should use zID, not student email, otherwise you can't attempt the multiple choice

Boolean Expressions

Programs that are Sensitive to Information

Up until now, our programs work top-down with no nuance — no section of our code is skipped, we cannot set aside a procedure only for a particular value, and we cannot check that our values are all 'valid'

Goal for Today

Create programs which allow us to give different answers depending on the input information.

Comparisons

- How do we differentiate between values? We need some notion of *comparison*
- We should be able to compare different values and ask some natural questions:
  - Is x larger than y?
  - Is x smaller than y?
  - Is x equal to y?
  - Is x **not** larger than y?
  - And so on...
- **Question:** What are the two possible answers I should get from asking these kinds of questions?

Boolean Operators

`>`	Greater than	`<`	Less than
`>=`	Greater than or equal to	`<=`	Less than or equal to
`==`	Equal to	`not`	Inverse

- The six boolean operators above should all be familiar to us
- All of them when used will evaluate to either `True` or `False`
- **Examples:**
  - `15 > 10` is `True`
  - `12 <= 10` is `False`
  - `"string" == "string"` is `True`
  - `not(15 > 10)` is `False`

Boolean Conjunctives

- We can chain together boolean statements using the conjunctives `and` or `or`
  - X `and` Y — returns `True` if and only if both $X =$ `True` and $Y =$ `True`
  - X `or` Y — returns `False` if and only if both $X =$ `False` and $Y =$ `False`
- **Questions:**
  - What does `(15 > 10) and (12 <= 10)` return?
  - What does `(15 > 10) or (12 <= 10)` return?
  - What does `("string" == "string") and (15 > 10)` return?

Distributing a `not` Over a Conjunctive

De Morgans' Laws

Let X and Y be boolean expressions. Then

$$\texttt{not}(X \texttt{ and } Y) = \texttt{not}(X) \texttt{ or } \texttt{not}(Y)$$

and

$$\texttt{not}(X \texttt{ or } Y) = \texttt{not}(X) \texttt{ and } \texttt{not}(Y).$$

- Take away: to distribute, push the `not` to each expression, and then swap every conjunctive
- This usually comes up implicitly in the MCQ this week

If/Elif/Else Statements

If Statements

- We can now use boolean expressions to control the flow of our program
- We do this using an `if` statement
  - Structure:

```
if boolean_expression is True:
    # do this
# move on
```

- The code that evaluates `if` the `boolean_expression` is `True` needs to be indented (4 spaces/1 tab).
- The rest of the code below is executed whether or not `boolean_expression` is `True` or `False`.

Examples of If Statements

- What is the value of x after this program terminates?

```
x = 15

if x > 10:
    x -= 5
x -= 5
```

- What is the value of y after this program terminates?

```
x = 15

y = 3
if x > 10 and y > 5:
    y = x
x -= 5
```

Isolated If-Statements

- An if-statement in isolation will still perform all its code below it after the indented block is complete
- What if we want to preclude the program from continuing with the rest of the code if it passes through the if-statement?

If/Else Statements

- We do this using the `if-else` keywords
  - Structure:

```
if boolean_expression is True:
    # do this
else:
    # do this instead
```

Examples of If/Else Statements

- What is the value of x after this program terminates?

```
x = 3

if x > 10:
    x -= 5
else:
    x += 5
```

- What is the value of y after this program terminates?

```
x = 10
y = 3

if x >= 10 and y > 5:
    y = x
else:
    y = -x
```

Isolated If/Else Statements

- Sometimes if-else statements are not enough to cover all the branches we want to go down — a continuation of conditions
- We can create more branches by using an elif statement

If/Elif/Else

- We can create an arbitrary number of branches by sandwiching `elif` between an `if` and an `else`
- Structure

```
if boolean_expression_one is True:
    # do this
elif boolean_expression_two is True:
    # do this instead
elif boolean_expression_three is True:
    # do this instead
...
else:
    # do this instead of all the above
```

Examples of If/Elif/Else Statements (I)

- What will this program print:

```
x = 15

if x < 10:
    print("Less than 10")
elif x < 20:
    print("Less than 20, not less than 10")
else:
    print("Greater than or equal to 20")
```

Examples of If/Elif/Else Statements (II)

- What will this program print:

```
x = 15

if x < 10:
    print("Less than 10")
elif x < 20:
    print("Less than 20, not less than 10")
elif x < 17:
    print("Less than 17, not less than 20, \
           not less than 10")
else:
    print("Greater than or equal to 20")
```

Constructing an If/Elif/Else Block

- Over the next few slides, we will build a program that takes a mark from the user and assigns a grade.
- We will aim to minimize repetition, redundancy, and errors as much as possible.

Attempt 1: Edge cases

- What happens if I put a grade of -2 below? or a grade of 200?

```
grade = float(input("What is your grade?"))

if grade < 50:
    print("fail")
elif grade < 85:
    print("distinction")
else:
    print("high distinction")
```

Attempt 2: Repetition

- How can the following be improved?

```
grade = float(input("What is your grade?"))

if grade < 0:
    print("haha nice one")
elif grade < 50:
    print("fail")
elif grade < 85:
    print("distinction")
else grade < 100:
    print("high distinction")
else:
    print("haha nice one")
```

Attempt 3: Redundancy

- Is every `or` / `and` necessary?

```
grade = float(input("What is your grade?"))

if grade < 0 or grade > 100:
    print("haha nice one")
elif grade >= 0 and grade < 50:
    print("fail")
elif grade >=50 and grade < 85:
    print("distinction")
elif grade >=85 and grade < 100:
    print("high distinction")
```

Final Design

```
grade = float(input("What is your grade?"))

if grade < 0 or grade > 100:
    print("haha nice one")

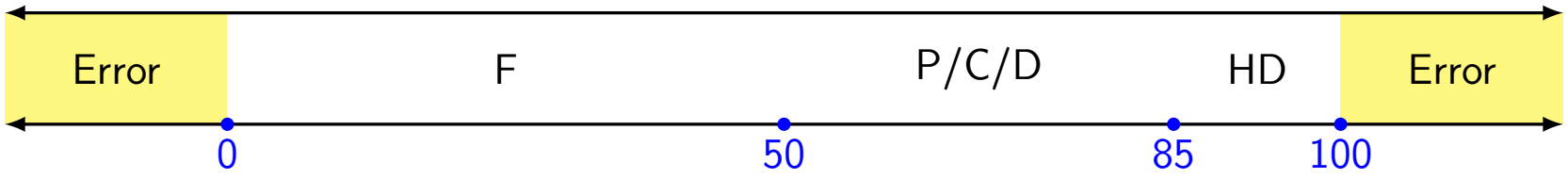
elif grade < 50:
    print("fail")

elif grade < 85:
    print("distinction")

else:
    print("high distinction")
```

On the next slides, we'll look at how this works visually.

Final Design




```
grade = float(input("What is your grade?"))

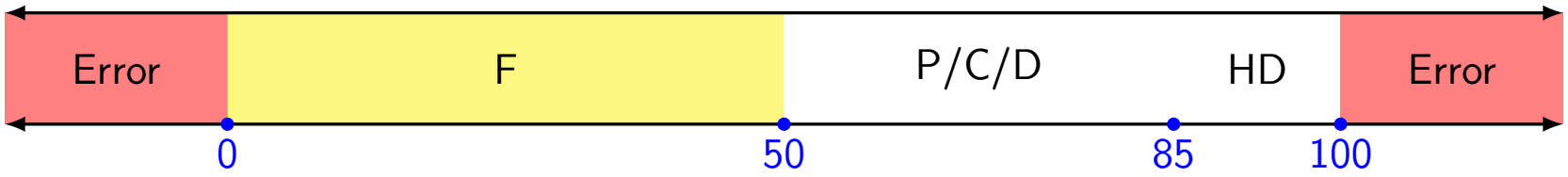
if grade < 0 or grade > 100:
    print("haha nice one")

elif grade < 50:
    print("fail")

elif grade < 85:
    print("distinction")

else:
    print("high distinction")
```

Final Design




```
grade = float(input("What is your grade?"))

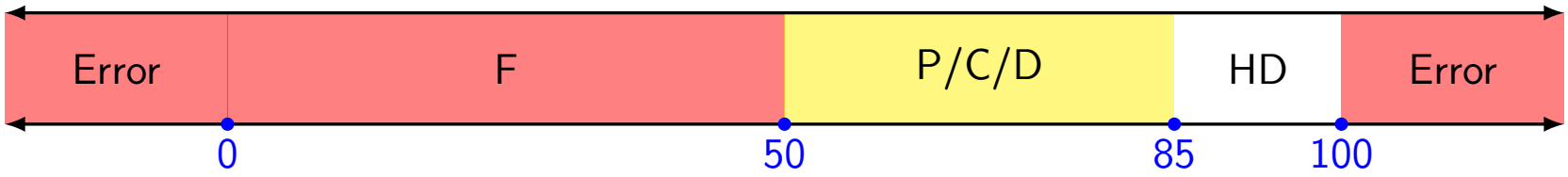
if grade < 0 or grade > 100:
    print("haha nice one")

elif grade < 50:
    print("fail")

elif grade < 85:
    print("distinction")

else:
    print("high distinction")
```

Final Design




```
grade = float(input("What is your grade?"))

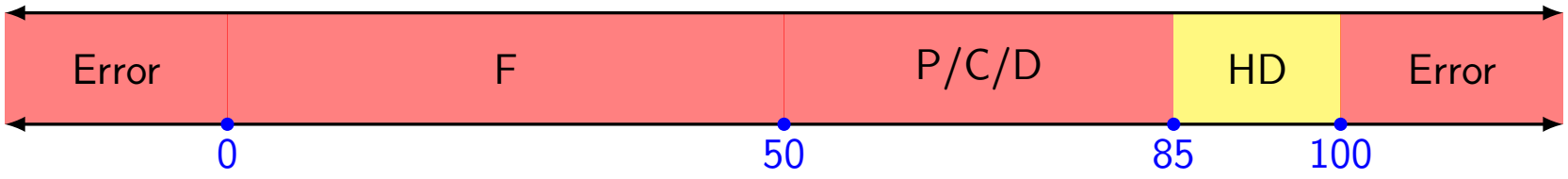
if grade < 0 or grade > 100:
    print("haha nice one")

elif grade < 50:
    print("fail")

elif grade < 85:
    print("distinction")

else:
    print("high distinction")
```

Final Design




```
grade = float(input("What is your grade?"))

if grade < 0 or grade > 100:
    print("haha nice one")

elif grade < 50:
    print("fail")

elif grade < 85:
    print("distinction")

else:
    print("high distinction")
```

Plotting

Plotting

- Python by itself cannot plot
- Need to use a library to help
- ```
  import matplotlib.pyplot as plt
  plt.figure()                    # Creates a new figure
  plt.plot(x_data, y_data)        # Adds x and y data
  plt.xlabel("X label")           # Label for x-axis
  plt.ylabel("Y label")           # Label for y-axis
  plt.title("Title")              # Adds a title
  ```
  ```

Demo: Using the matplotlib Library

- Here, we will walk through an example of how to plot using matplotlib.
- You will need to plot some bearings and their corresponding angles in Exercise 1, which can be easily adapted from the format in the demo.

Lab Tips

Lab Tips

- Exercise 1
  - You can use any method/programs to work out the equations of the two lines
  - Your script needs to work for decimal numbers as well, not just integers
  - Bearing of $270°$ corresponds to an angle of $+180°$, not $-180°$
- Exercise 2
  - Try and make sure that the checks you make in subsequent aren't redundant
- Exercise 3
  - Go from top to bottom, or bottom to top layers

Feedback

Feel free to provide anonymous feedback about the lab!

Feedback Form