# Every Boilermaker Engineer Codes: 101 Entry-Level Programming in Python

## Lecture 02

Dr. John H. Cole

`<jhcole@purdue.edu>`

# PURDUE
U N I V E R S I T Y ®

COLLEGE OF ENGINEERING

Fall 2022

- Presentation type 'e' or 'E' indicates exponential notation

### Terminal

```
>>> format(123456.789, 'e')
'1.234568e+05'
>>> format(123456.789, '.2e')
'1.23e+05'
>>> format(0.00000123456, '.4e')
'1.2346e-06'
>>> format(0.00000123456, '.4E')
'1.2346E-06'
```

### MATH EQUIVALENT

$1.234,568 \times 10^5$

$1.23 \times 10^5$

$1.2346 \times 10^{-6}$

$1.2346 \times 10^{-6}$

# Part I

## Logic and Decision Structures

## NONETYPE VALUES

- None is the only value of the type NoneType
- represents the absence of a value
- frequently used when
  - default arguments are not passed to a function
  - default return value when a function returns nothing
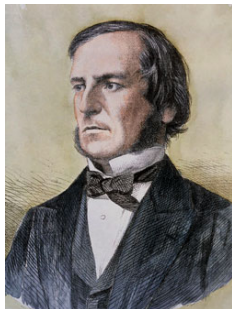
---

Terminal

```
>>> None
>>> type(None)
<class 'NoneType'>
>>> a = print('Hello')
Hello
>>> type(a)
<class 'NoneType'>
>>> print(a)
None
```

## Boolean Values

- either True or False and nothing else
- are of type 'bool'
- named after George Boole (1815-1864)
- commonly used as flags to signal when a condition exists
  - set to True $\implies$ condition exist
  - set to False $\implies$ condition does not exist

### Terminal

```
>>> True
True
>>> False
False
>>> type(True)
<class 'bool'>
>>> type(False)
<class 'bool'>
```



George Boole, c. 1860

## The bool() Function

- converts a value into a bool from any other type

Terminal

```
>>> bool(0)
False
>>> bool(0.0)
False
>>> bool('')
False
>>> bool(False)
False
>>> bool(None)
False
```

Terminal

```
>>> bool(-5)
True
>>> bool(0.00000000001)
True
>>> bool('    ')
True
>>> bool(True)
True
>>>
```

## LOGICAL OPERATORS

BOOLEAN EXPRESSION an expression that results in a bool
LOGICAL OPERATOR used to create complex Boolean
expressions
NOT unary operator, reverses its operand
AND binary operator, true if both operands
are true
OR binary operator, true if either operand
is true

| A | not A |
|---|---|
| False | True |
| True | False |

| A | B | A and B |
|---|---|---|
| False | False | False |
| False | True | False |
| True | False | False |
| True | True | True |

| A | B | A or B |
|---|---|---|
| False | False | False |
| False | True | True |
| True | False | True |
| True | True | True |

## LOGICAL OPERATORS EXAMPLES

- order of operations is determined by precedence
- higher precedence operators execute first

1 parentheses ()
2 not
3 and
4 or

### Terminal

```
$ python
>>> True and not False
True
>>> not (False and not False)
True
>>> True or False and False
True
>>> True or True and False
True
>>> (not False or False) and False
False
>>> not False or False and False
True
```

## SHORT-CIRCUIT EVALUATION

Determining the result of an and or or operator after evaluating only one of its operands

- and is False if its left operand is False
- or is True if its left operand is True

| Terminal |
| --- |
| ```
>>> False and True
False
>>> False and False
False
>>> False and undefined
False
>>> False and print('hi')
False
>>>
``` |

| Terminal |
| --- |
| ```
>>> True or True
True
>>> True or False
True
>>> True or undefined
True
>>> True or print('hi')
True
>>>
``` |

## Relational Operators

Relational Operator   returns a bool indicating if a specific
relationship exists between its operands

- equality '=='
    - different from the assignment operator '='
    - back-to-back '=', no space
- not equal '!='
- greater than '>'
- greater than or equal to '>='
- less than '<'
- less than or equal to'<='

## RELATIONAL OPERATORS - NUMERIC EXAMPLES

Terminal

```
$ python
>>> 5 == 5
True
>>> 5 == 3
False
>>> 5 != 5
False
>>> 5 != 3
True
```

Terminal

```
>>> 5 > 3
True
>>> 5 > 5
False
>>> 5 >= 5
True
>>> 3 >= 5
False
>>>
```

## Numeric Range Checking

- use and to determine if a value is *within* a range
- use or to determine if a value is *outside* of a range

### Terminal

```
>>> x = 20
>>> x < 10 or 20 < x
False
>>> x < 10 or 20 <= x
True
>>> 10 < x and x < 20
False
>>> 10 <= x and x <= 20
True
>>> 10 <= x <= 20
True
```

### Math Equivalent

$\{x | (-\infty, 10) \cup (20, \infty)\}$

$\{x | (-\infty, 10) \cup [20, \infty)\}$
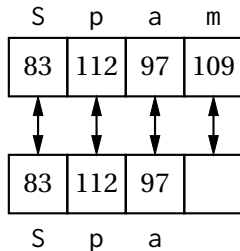
$\{x | (10, 20)\}$

$\{x | [10, 20]\}$

$\{x | [10, 20]\}$

## RELATIONAL OPERATORS WITH STRINGS

- strings are compared character by character based on their ASCII values
- 'A' < 'Z', 'Z' < 'a', 'a' < 'z'

ASCII Table

| Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char | Dec | Char |
|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|-----|------|
| 0 | NUL | 16 | | 32 | SPACE | 48 | 0 | 64 | @ | 80 | P | 96 | ' | 112 | p |
| 1 | | 17 | | 33 | ! | 49 | 1 | 65 | A | 81 | Q | 97 | a | 113 | q |
| 2 | | 18 | | 34 | " | 50 | 2 | 66 | B | 82 | R | 98 | b | 114 | r |
| 3 | | 19 | | 35 | # | 51 | 3 | 67 | C | 83 | S | 99 | c | 115 | s |
| 4 | | 20 | | 36 | $ | 52 | 4 | 68 | D | 84 | T | 100 | d | 116 | t |
| 5 | | 21 | | 37 | % | 53 | 5 | 69 | E | 85 | U | 101 | e | 117 | u |
| 6 | | 22 | | 38 | & | 54 | 6 | 70 | F | 86 | V | 102 | f | 118 | v |
| 7 | | 23 | | 39 | ' | 55 | 7 | 71 | G | 87 | W | 103 | g | 119 | w |
| 8 | BS | 24 | | 40 | ( | 56 | 8 | 72 | H | 88 | X | 104 | h | 120 | x |
| 9 | TAB | 25 | | 41 | ) | 57 | 9 | 73 | I | 89 | Y | 105 | i | 121 | y |
| 10 | LF | 26 | | 42 | * | 58 | : | 74 | J | 90 | Z | 106 | j | 122 | z |
| 11 | | 27 | ESC | 43 | + | 59 | ; | 75 | K | 91 | [ | 107 | k | 123 | { |
| 12 | | 28 | | 44 | , | 60 | < | 76 | L | 92 | \ | 108 | l | 124 | | |
| 13 | | 29 | | 45 | - | 61 | = | 77 | M | 93 | ] | 109 | m | 125 | } |
| 14 | | 30 | | 46 | . | 62 | > | 78 | N | 94 | ^ | 110 | n | 126 | ~ |
| 15 | | 31 | | 47 | / | 63 | ? | 79 | O | 95 | _ | 111 | o | 127 | DEL |

## RELATIONAL OPERATORS - STRING EXAMPLES

'Spam' > 'Spa'

| S | p | a | m |
|----|-----|----|-----|
| 83 | 112 | 97 | 109 |

| 83 | 112 | 97 | |
|----|-----|----|---|
| S | p | a | |

- strings are compared character by character based on their ASCII values
- comparisons are case sensitive
- when characters are equal, longer strings are greater than shorter stings

## RELATIONAL OPERATORS - STRING VALIDATION

What if user enters unexpected input?

```
Terminal

>>> ans = input('Please enter Yes or No: ')
Please enter Yes or No: yes
>>> ans
'yes'
>>> ans == 'Yes'
False
>>> ans == 'YES'
False
>>> ans == 'Yes' or ans == 'YES' or ans == 'yes'
True
>>>
```

Thanks for watching!

## CONTROL STRUCTURES

Control structures determine the order in which a set of statements are executed.

SEQUENCE STRUCTURE default, statements that execute in the order they appear

CONDITIONAL STRUCTURE statements execute only if a condition is met

- also known as selection structure, or decision structure

REPETITION STRUCTURE statements execute repeatedly until some condition is met

- also known as loop structure, or iteration structure

## CONDITIONAL STRUCTURE

- often portrayed using flow charts (diamond represents a condition test)
- basic form provides only one alternative path of execution
- implemented in code via the if statement

## THE if STATEMENT

- 'if condition:' is known as the if clause
- condition evaluates to either True or False
- the if clause ends with a colon ':'
- indented statement block only executes if condition is True

### Editor - if_syntax.py

```python
1  statement_1
2
3  if condition:
4      statement_2
5      statement_3
6
7  statement_4
```
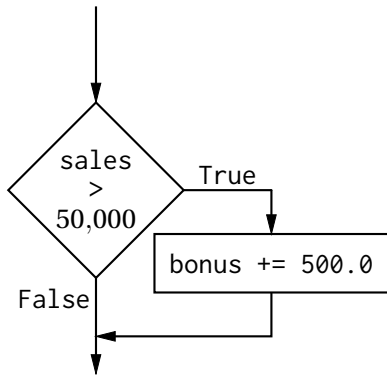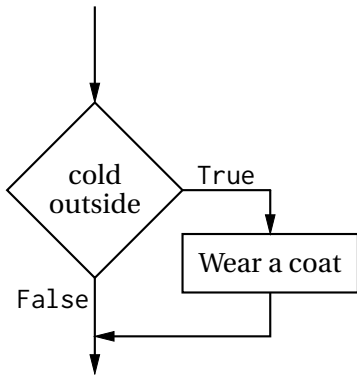
statement_1  always executes first

statement_2  executes second only if condition is True

statement_3  executes third only if condition is True

statement_4  always executes last

## THE if STATEMENT (CONT.)

- statements in the same block must be indented equally

### Editor - if_example.py

```
1 a, b = 5, 3
2 if a > b:
3     print('a is more')
4
5 print('I always run')
6 if b > a:
7     print('b is more')
8     print('a is less')
9
10 print("I'm finished")
```

- indent with tabs or spaces but not both
- convention is four spaces
- blank lines are ignored

### Terminal

```
$ python if_example.py
a is more
I always run
I'm finished
$
```

## NESTED if STATEMENTS

- if statements can be nested by indenting more
- proper indentation is required by Python interpreter
- makes code more readable for humans too

Editor - nested_if.py

```python
1 a, b = 5, 3
2 if a == 5:
3     print('a is 5')
4     if b == 4:
5         print('b is 4')
6     print('spam')
7 print("I'm finished")
```

Terminal

```
$ python nested_if.py
a is 5
spam
I'm finished
$
```

## THE if-else STATEMENT

- dual alternative decision structure provides two alternatives
- implemented in code via the if-else statement
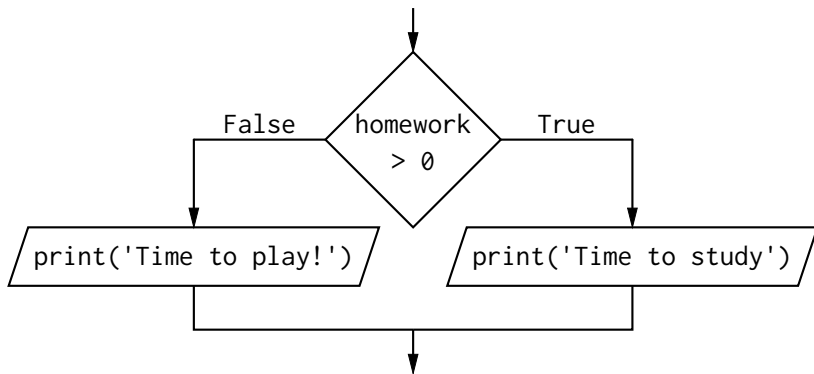
# THE if-else STATEMENT (CONT.)

- one of two alternative will be chosen
- the else clause ends with a colon ':'
- else block should align with matching if block

### Editor - if_else.py

```python
print('How many ', end='')
hw = input('assignments? ')

if int(hw) > 0:
    print('Time to study.')
else:
    print('Time to play!')
print('Time to sleep.')
```

### Terminal

```
$ python if_else.py
How many assignments? 3
Time to study.
Time to sleep.
$ python if_else.py
How many assignments? 0
Time to play!
Time to sleep.
```

## if-else Example

### Editor - loan.py

```python
1  # This program determines whether a bank customer
2  # qualifies for a loan.
3  MIN_SALARY = 30000.0 # The minimum annual salary
4  MIN_YEARS = 2 # The minimum years on the job
5  # Get the customer's annual salary.
6  salary = float(input('Enter your annual salary: '))
7  # Get the number of years on the current job.
8  job_years = int(input('Enter the number of ' +
9      'years employed: '))
10 # Determine whether the customer qualifies.
11 if salary >= MIN_SALARY and job_years >= MIN_YEARS:
12     print('You qualify for the loan.')
13 else:
14     print('You do not qualify for this loan.')
```

# NESTED if-else
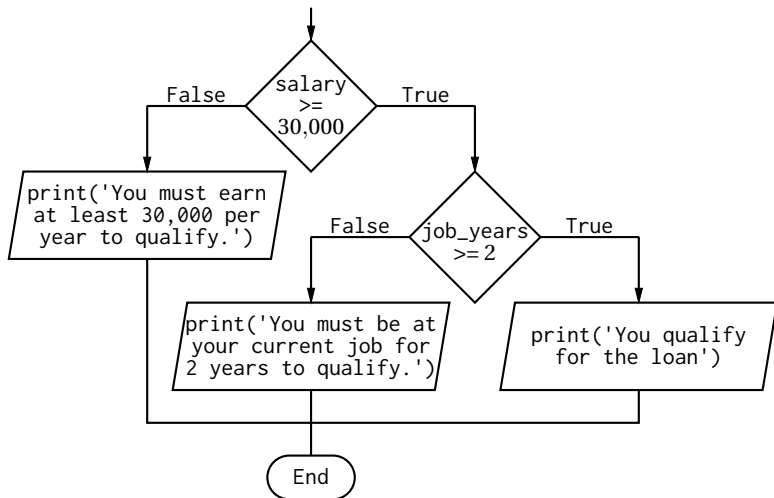
## NESTED if-else (CONT.)

- proper indentation is required by Python interpreter

```
Editor - loan_nested_if_else.py

 9  --snip--
10  # Determine whether the customer qualifies.
11  if salary >= MIN_SALARY:
12      if job_years >= MIN_YEARS:
13          print('You qualify for the loan.')
14      else:
15          print('You must be at your current job',
16              f'for {MIN_YEARS} years to qualify.')
17  else:
18      print('You must earn at least',
19          f'{MIN_SALARY:,.0f} per year to qualify.')
```

## NESTED if-else REFACTORED

```
Editor - loan_refactored.py

9   --snip --
10  # Determine whether the customer qualifies.
11  if salary < MIN_SALARY:
12      print('You must earn at least',
13            f'{MIN_SALARY:,.0f} per year to qualify.')
14  else:
15      if job_years < MIN_YEARS:
16          print('You must be at your current job',
17                f'for {MIN_YEARS} years to qualify.')
18      else:
19          print('You qualify for the loan.')
```
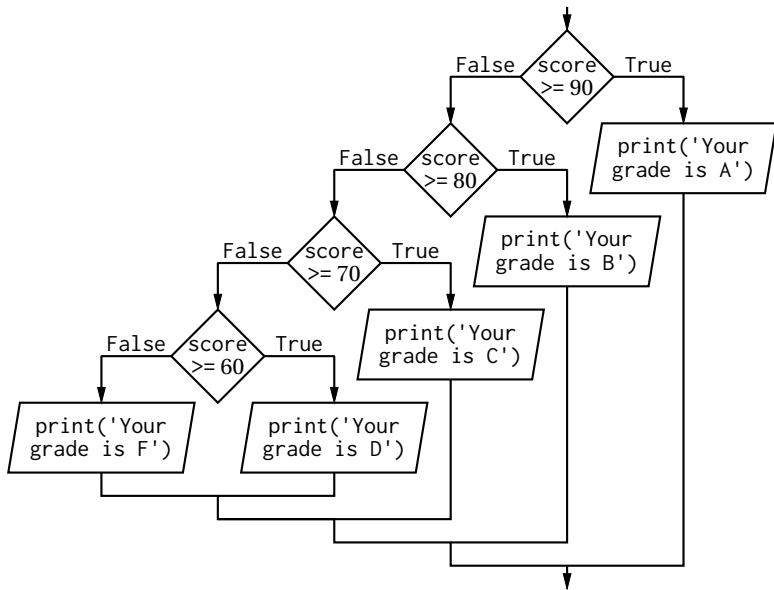
# THE if-elif-else STATEMENT

ELIF  a special version of a decision structure

- all clauses aligned, all statement blocks aligned

---

Editor - loan_if_elif_else.py

```
 9   --snip --
10  # Determine whether the customer qualifies.
11  if salary < MIN_SALARY:
12      print('You must earn at least',
13            f'{MIN_SALARY:,.0f} per year to qualify.')
14  elif job_years < MIN_YEARS:
15      print('You must be at your current job',
16            f'for {MIN_YEARS} years to qualify.')
17  else:
18      print('You qualify for the loan.')
```

# THE if-elif-else STATEMENT

## THE if-elif-else STATEMENT

### Editor - grade_if_elif_else.py

```python
1  # This program gets a numeric test score from the
2  # user and displays the corresponding letter grade.
3
4  # Variables to store the grade thresholds.
5  A_CUTOFF = 90
6  B_CUTOFF = 80
7  C_CUTOFF = 70
8  D_CUTOFF = 60
9
10 # Get a test score from the user.
11 score = int(input('Enter your test score: '))
12
13   --snip--
```

## THE if-elif-else STATEMENT

```
Editor - grade_if_elif_else.py

12  --snip--
13  # Determine the grade.
14  if score >= A_CUTOFF:
15      print('Your grade is A.')
16  elif score >= B_CUTOFF:
17      print('Your grade is B.')
18  elif score >= C_CUTOFF:
19      print('Your grade is C.')
20  elif score >= D_CUTOFF:
21      print('Your grade is D.')
22  else:
23      print('Your grade is F.')
```

## DECISION EXAMPLES

### Editor - decisions.py

```python
1  x = 12
2  if x > 5:
3      print('x>5')
4      print('greater than 5')
5  if x > 9:
6      print('greater than 9')
7  elif x == 12:
8      print('equal to 12')
9  if x%3:
10     print('not divisible by 3')
11 elif x%4:
12     print('not divisible by 4')
13 else:
14     print('neither')
```

### Terminal

```
$ python decisions.py
x > 5
greater than 5
greater than 9
neither
$
```

## PRACTICE

Write a program that asks the user for a number in the range of 1 through 7. The program should display the corresponding day of the week, where 1 = Monday, 2 = Tuesday, 3 = Wednesday, 4 = Thursday, 5 = Friday, 6 = Saturday, 7 = Sunday. The program should display an error message if the user enters a number that is outside the range of 1 through 7.

## Practice

### Editor - day_of_week.py

```python
1 # This program gets a weekday number from the user
2 # and displays the corresponding weekday name.
3
4 # Get the number for the day of the week
5 day = int(input('Enter a number (1-7) for'
6                 + ' the day of the week: '))
7
8 # Determine the name of the day of the
9 # week, and display it.
10   --snip--
```

## Practice

#### Editor - day_of_week.py

```python
10  if day == 1:
11      print('Monday')
12  elif day == 2:
13      print('Tuesday')
14  elif day == 3:
15      print('Wednesday')
16  elif day == 4:
17      print('Thursday')
18  elif day == 5:
19      print('Friday')
20  elif day == 6:
21      print('Saturday')
22  elif day == 7:
23      print('Sunday')
24  else:
25      print('Please enter a',
26            'number in the',
27            'range 1-7.')
```

#### Editor - wrong.py

```python
10  if day == 1:
11      print('Monday')
12  if day == 2:
13      print('Tuesday')
14  if day == 3:
15      print('Wednesday')
16  if day == 4:
17      print('Thursday')
18  if day == 5:
19      print('Friday')
20  if day == 6:
21      print('Saturday')
22  if day == 7:
23      print('Sunday')
24  else:
25      print('Please enter a',
26            'number in the',
27            'range 1-7.')
```

# Thanks for watching!

# Part II

## Homework Tips

## Calculating the Days, Hours, Minutes and Seconds

Terminal

```
$ python3
>>> time = 810549
>>> days = time // 86400
9
>>> time - days*86400
32949
>>> time % 86400
32949
>>> hours = time % 86400 // 3600
9
>>> time % 3600
549
>>> minutes = time % 3600 // 60
9
```

# GETTING THE OUTPUT RIGHT: TIP 1

Editor - tip1.py

```python
print(f'  {time} seconds is: ', end = '')
if d:
    print(f'{d} day(s)', end = '')
if h:
    print(f'{h} hours(s)', end = '')
if m:
    print(f'{m} minutes(s)', end = '')
if s:
    print(f'{s} seconds(s)', end = '')
print('.')
```

# GETTING THE OUTPUT RIGHT: TIP 2

Editor - tip2.py

```python
1  if h:
2      if d:
3          if m or s:
4              print(', ', end = '')
5          else:
6              print(' and ', end = '')
7      print(f'{h} hours(s)', end = '')
```