Every Boilermaker Engineer Codes: 101
Entry-Level Programming in Python
Lecture 02

Dr. John H. Cole
<jhcole@purdue.edu>

PURDUE
UNIVERSITY®

COLLEGE OF ENGINEERING

Spring 2021

Exponential Notation Format

- 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

$1.234{,}568 \times 10^{5}$

1.23×10^{5}

1.2346×10^{-6}

1.2346×10^{-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

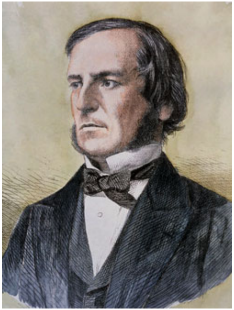
```
>>> None
>>> type(None)
<class 'NoneType'>
>>> a = print('Hello')
Hello
>>> type(a)
<class 'NoneType'>
>>> print(a)
None
```

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` ⟹ condition exist
  - set to `False` ⟹ 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

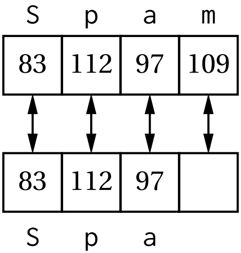
- strings are compared character by character based on their ASCII values
- ‘A’ < ‘Z’, ‘Z’ < ‘a’, ‘a’ < ‘z’

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	ˆ	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

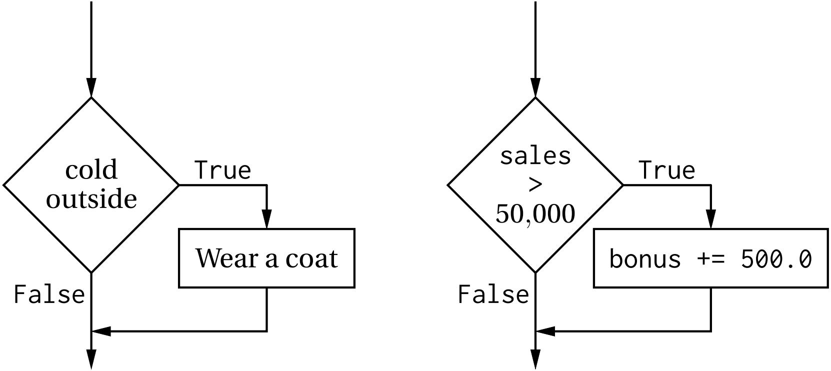
- also known as selection structure, or decision structure

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

```
statement_1

if condition:
    statement_2
    statement_3

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

```
a, b = 5, 3
if a > b:
    print('a is more')

print('I always run')
if b > a:
    print('b is more')
    print('a is less')

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

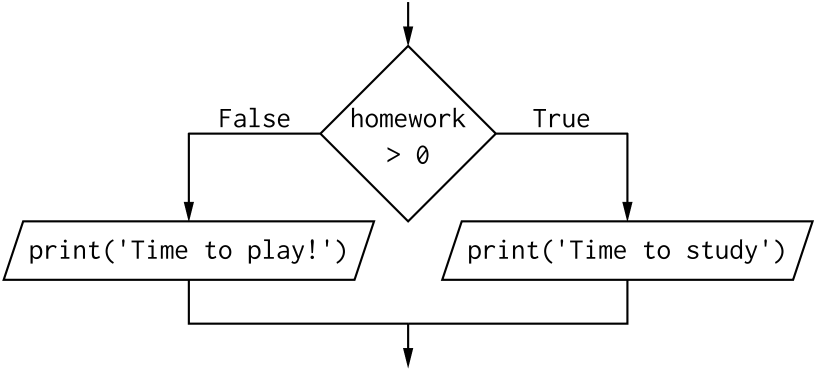
```python
a, b = 5, 3
if a == 5:
    print('a is 5')
    if b == 4:
        print('b is 4')
    print('spam')
print("I'm finished")
```

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

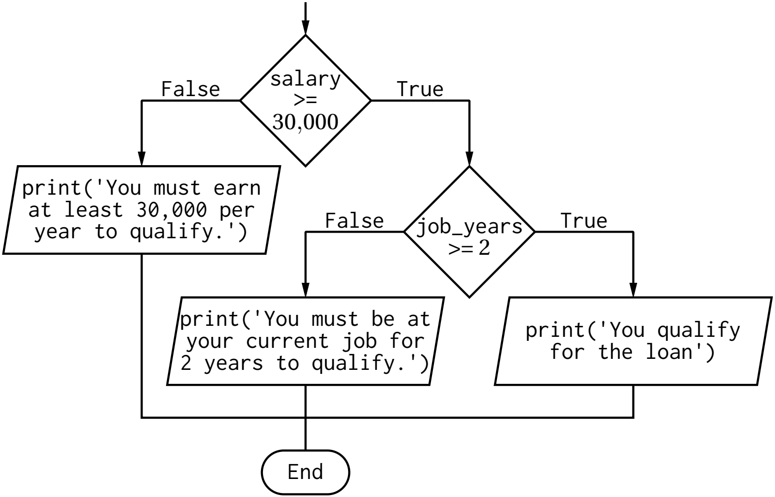
```
$ python if_else.py
How many assignments? 3
Time to study.
Time to sleep.
$ python if_else.py
How many assignments? 0
Time to play!
Time to sleep.
```

if-else Example

Editor - loan.py

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

Nested `if-else`


salary >= 30,000
False
True
print('You must earn at least 30,000 per year to qualify.')
job_years >= 2
False
True
print('You must be at your current job for 2 years to qualify.')
print('You qualify for the loan')
End

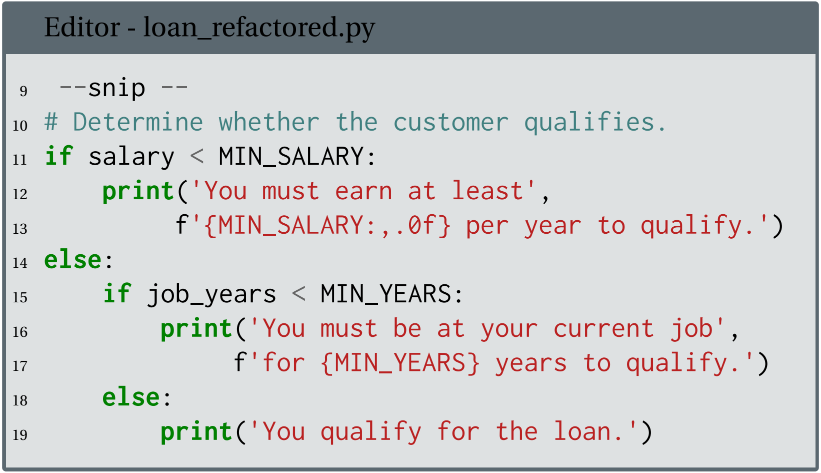
NESTED if-else (CONT.)

- proper indentation is required by Python interpreter

Editor - loan_nested_if_else.py

```python
--snip--
# Determine whether the customer qualifies.
if salary >= MIN_SALARY:
    if job_years >= MIN_YEARS:
        print('You qualify for the loan.')
    else:
        print('You must be at your current job',
              f'for {MIN_YEARS} years to qualify.')
else:
    print('You must earn at least',
          f'{MIN_SALARY:,.0f} per year to qualify.')
```

Nested if-else Refactored



Editor - loan_refactored.py

```python
 --snip --
# Determine whether the customer qualifies.
if salary < MIN_SALARY:
    print('You must earn at least',
         f'{MIN_SALARY:,.0f} per year to qualify.')
else:
    if job_years < MIN_YEARS:
        print('You must be at your current job',
             f'for {MIN_YEARS} years to qualify.')
    else:
        print('You qualify for the loan.')
```

The if-elif-else Statement

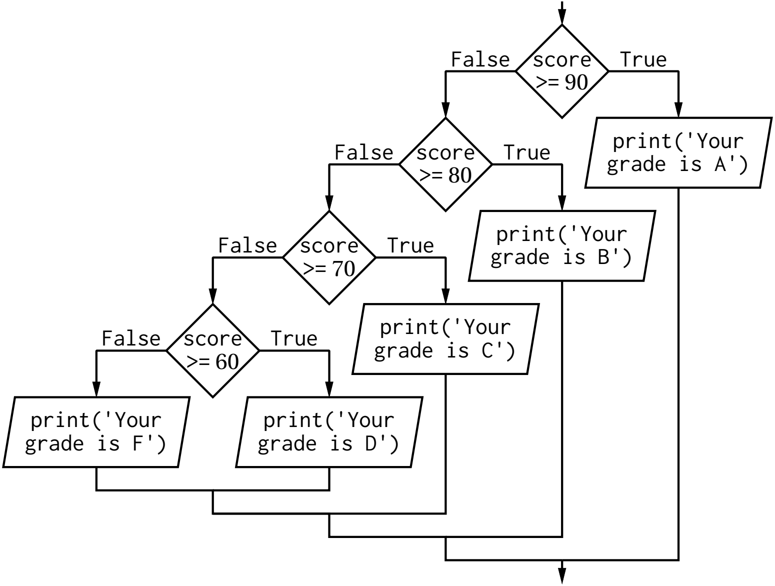
ELIF a special version of a decision structure

- all clauses aligned, all statement blocks aligned

Editor - loan_if_elif_else.py

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

THE if-elif-else STATEMENT



The if-elif-else Statement

Editor - grade_if_elif_else.py

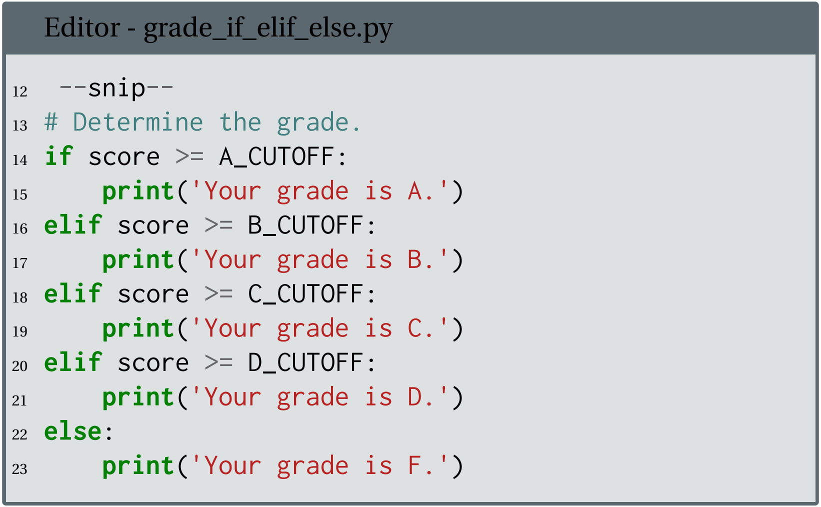
```python
# This program gets a numeric test score from the
# user and displays the corresponding letter grade.

# Variables to store the grade thresholds.
A_CUTOFF = 90
B_CUTOFF = 80
C_CUTOFF = 70
D_CUTOFF = 60

# Get a test score from the user.
score = int(input('Enter your test score: '))

 --snip--
```

The if-elif-else Statement



Editor - grade_if_elif_else.py

```
 --snip--
# Determine the grade.
if score >= A_CUTOFF:
    print('Your grade is A.')
elif score >= B_CUTOFF:
    print('Your grade is B.')
elif score >= C_CUTOFF:
    print('Your grade is C.')
elif score >= D_CUTOFF:
    print('Your grade is D.')
else:
    print('Your grade is F.')
```

Decision Examples

Editor - decisions.py

```python
x = 12
if x > 5:
    print('x>5')
    print('greater than 5')
if x > 9:
    print('greater than 9')
elif x == 12:
    print('equal to 12')
if x%3:
    print('not divisible by 3')
elif x%4:
    print('not divisible by 4')
else:
    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

```
# This program gets a weekday number from the user
# and displays the corresponding weekday name.

# Get the number for the day of the week
day = int(input('Enter a number (1-7) for',
                'the day of the week: '))

# Determine the name of the day of the
# week, and display it.
 --snip--
```

Practice

Editor - day_of_week.py

```python
if day == 1:
    print('Monday')
elif day == 2:
    print('Tuesday')
elif day == 3:
    print('Wednesday')
elif day == 4:
    print('Thursday')
elif day == 5:
    print('Friday')
elif day == 6:
    print('Saturday')
elif day == 7:
    print('Sunday')
else:
    print('Please enter a',
          'number in the',
          'range 1-7.')
```

Editor - wrong.py

```python
if day == 1:
    print('Monday')
if day == 2:
    print('Tuesday')
if day == 3:
    print('Wednesday')
if day == 4:
    print('Thursday')
if day == 5:
    print('Friday')
if day == 6:
    print('Saturday')
if day == 7:
    print('Sunday')
else:
    print('Please enter a',
          'number in the',
          'range 1-7.')
```

Thanks for watching!

Part II

Homework Tips

Calculating the Days, Hours, Minutes and Seconds

Terminal

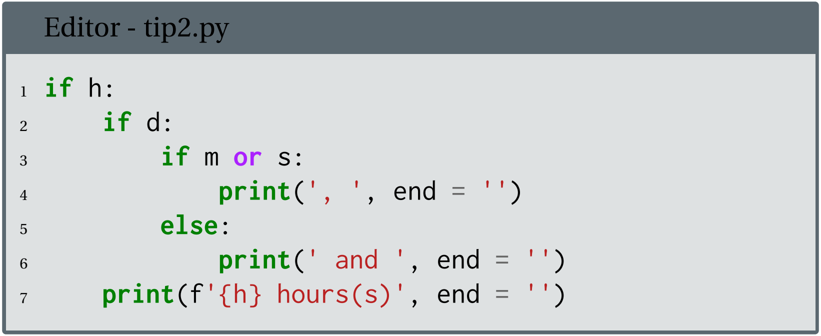
```
$ python3
>>> time = 810549
>>> days = time // 86400
9
>>> time - days*86400
32949
>>> time % 86400
32949
>>> hours = time % 86400 // 3600
9
>>> time % 3600
549
>>> minutes = time % 3600 // 60
9
```

Getting the Output Right: Tip 1

Editor - tip1.py

```
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
if h:
    if d:
        if m or s:
            print(', ', end = '')
        else:
            print(' and ', end = '')
    print(f'{h} hours(s)', end = '')
```