Class 4 (sec 2) - string formatting and conditional statements

Friday, January 26, 2024 9:13 AM

- Please put away your phones into your bags - if you need to use it, please step outside, then come back.
- When you have a **question**, feel free to **interrupt me**
- When you have an **answer**, please **raise your hand**

- Practice Problem solutions are to Brightspace
  Solution to HW1 will be sent out - but **strictly confidential, only for your use.**
- Practice Problem solutions to Brightspace
- Read Assignment 2 and think of / prepare test cases

Do you have any questions?

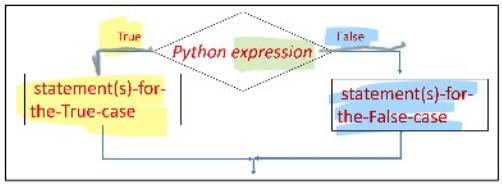
Handout 2 CS230 – Introduction to Programming with Python–Spring'24 Page 1 of 4

Handout 2

Selection and conditional statements branching

The **if** statement is used for conditional execution. Most frequently used form is

```
if expression:
    statement(s)-for-the-True-case
else:
    statement(s)-for-the-False-case
```



Examples: notice the indentation:

1.

```
if i > 0:
print("i is positive")
```

(a) Wrong

```
if i > 0:
    print("i is positive")
```

(b) Correct

2.

```
# Determine the interest rate, depending on the size of the
investment
if invest >= 1000:
    interestRate = 3.25
else:
    interestRate = 3.0

# calculate the ending balance
endBalance = invest * (1 + interestRate/100)
```

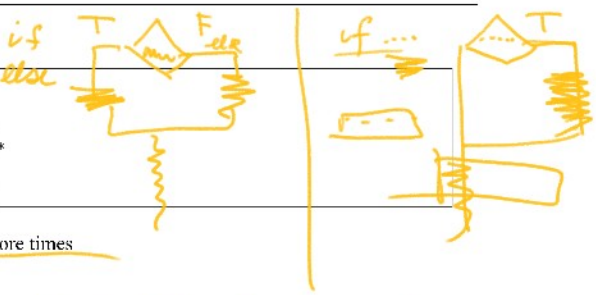
3.

```
# What values of time and limit will cause bonus to be 50? 100? 0?
if time < limit:
    print("You made it.")
    if (limit - time >= 10):
        bonus = 100;
    else:
        bonus = 50;
else:
    print("You missed the  deadline.")
    bonus = 0
```

time	7	5	10
limit	10	20	5
bonus	50	100	∅

GENERAL FORM

```
if expression :
     statements1
(elif expression:
     statements2)*
[else:
     statements3]
```



Here, * means repeated 0 or more times
[] means - optional

```
score = float(input ("Enter grade in the 0..100 range"))

if score >= 90:
    grade = 'A'
elif score >= 80:
    grade = 'B'
elif score >= 70:
    grade = 'C'
elif score >= 60:
    grade = 'D'
else:
    grade = 'F'
```

PRACTICE PROBLEMS

1. User will enter an investment value. Write a conditional statement that assigns interest rate based on investment value entered by the user, according to the following table

Investment range	Interest rate
less than $1,000	3.0%
$1,000 - $4,999	3.2%
$5,000 - $9,999	3.4%
$10,000 and above	3.6%

2. User will enter hour and minute values of start of call, then hour, minute of end of call. Check that the time of end of is actually past the time of start. Compute how long the call lasted and print out how many full 15-minute intervals fit within that time.

COMPARISONS

Numbers and strings can be compared using **Relational Operators**: also called Comparison Operators include: `>` `<` `>=` `<=` `==` `!=`
Result is a Boolean value `True` or `False`

Comparison of strings is done based on the order of the characters in the ASCII/UNICODE values.

```
''' comparison operators '''

print ('---comparison of numbers----------------------------')
i = 5; j = 9;
print (i > j);
print(i == j);
print(i != j);

print ('---comparison of Strings----------------------------')
str1, str2 = 'ABC', 'Foo'
print (str1 < str2)
print ('abc'<'abcd')
print ('ABC' < 'abc')
print ('ABC' == 'abc')
```

TRUE AND FALSE IN PYTHON

In the context of Boolean operations, and also when expressions are used by control flow statements.
The following values are interpreted as **`False`**:

- **`False`, `None`**, numeric **zero** of all types, and **empty** strings and **empty** containers (including strings, tuples, lists, dictionaries, sets).

All values not interpreted as **`False`**, are interpreted as **`True`**

BOOLEAN OPERATIONS: not, and, or

Used to combine results of multiple tests

In the **order of precedence** in evaluation, assuming a, b are expressions:

`not a`	Negation of a
`a and b`	True if both a and b are True-equivalent, and False otherwise
`a or b`	True if at least one of a or b is True-equivalent, and False otherwise

PRACTICE PROBLEMS

3. User will enter year in which they started college and year they graduated from college. Output *included pandemic years*, if the study period included the years of the COVID pandemic: 2020 and 2021.

4. User will enter time of entry and exit in/from the garage in hours and minutes, military time, as shown:

```
Please enter hour and minute of entry, first, hour: 12
then, minute: 45
Please enter hour and minute of exit, first, hour: 14
then, minute: 5
```

The charge for parking is
   a) $2, for less than 30 minutes stay at any time of the day, otherwise
   b) $15, if time of entry was between 6:00 and 7:30 am, and time of exit between 14:30 and 16:00,
   c) $5, per each full hour, for all other situations.

Given the time the car entered a garage and the time it exited, calculate the amount due.

Hint: to work with time values, convert hours and minutes into a minute-based representation. For example, the time of entry `12:45` entered above, converted to minutes is 765, which is `12*60 + 45`

CONDITIONAL OPERATOR

Syntax:

```
expressionTrue if condition else expressionFalse
```

Shorthand for:

```
if condition:
    expressionTrue
else:
    expressionFalse
```

Example:
The three statements below produce identical output

```
1. if num % 2 == 0:
       print ("even")
   else:
       print ("odd")

2. print("even") if num % 2    0 else print ("odd")

3. val = "even" if num % 2 == 0 else "odd"
   print(val)

4. print("even" if  num % 2 == 0  else "odd" )
```