ch_5_assignment

March 14, 2023

Ch_5_assignment

[ ]:
```
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = 'all'
```

1 Making Choices

1.1 A Boolean Type

1.1.1 Boolean Operato

[ ]:
```
# There are only three basic Boolean operators: and, or, and not. not has the␣
 ↪highest precedence, followed
# by and, followed by or.
not True
not False
print()
True and True
False and False
True and False
False and True
print()
True or True
False or False
True or False
False or True
```

[ ]: False

[ ]: True

[ ]: True

[ ]: False

[ ]: False

[ ]: False

[ ]: True

[ ]: False

[ ]: True

[ ]: True

```
cold = True
windy = True
(not cold) and windy
not (cold and windy)
```

[ ]: False

[ ]: False

cold	windy	cold and windy	cold or windy	(not cold) and windy	not(cold and windy)
True	True	True	True	False	False
True	False	False	True	Fasle	True
True	True	Fasle	True	True	True
False	False	False	False	False	True

1.1.2 Relational Operators

Symnol	Operation
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to
==	Equal to
!=	Not equal to

[ ]:
```
#In this docstring, we use the acronym "iff," which stands for "if and only if.
 ↪" An equivalent phrase is "exactly when."
def is_positive(x:float)->bool:
    return x > 0
is_positive(3)
```

```
is_positive(-4.6)
is_positive(0)
```

[ ]: True

[ ]: False

[ ]: False

1.1.3 Combining Comparison

[ ]:
```
# Arithmetic Operators : +, - so on
# Boolean Operators: and, or, and not
# Relational Operators: <, ==, and so o
x = 2
y = 5
z = 7
x < y and y < z
```

[ ]: True

[ ]:
```
# Arithmetic operators have higher precedence than relational operators.
# Relational operators have higher precedence than Boolean operators.
# All relational operators have the same precedence.
x = 3
(1 < x) and (x <= 5)

x = 7
(1 < x) and (x <= 5)

x = 3
1 < x <= 5
```

[ ]: True

[ ]: False

[ ]: True

[ ]:
```
3 < 5 != True

3 < 5 != False
```

[ ]: True

[ ]: True

1.1.4 Short-Circuit Evaluation

[ ]:
```
1 / 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
Cell In[17], line 1
----> 1 1 / 0

ZeroDivisionError: division by zero
```

[ ]:
```
(2 < 3) or (1 / 0)
```

[ ]: True

1.1.5 Comparing Strings

[ ]:
```
# The characters in strings are represented by integers: a capital A, for␣
 ↪example, is represented by 65, whereas a space is 32, and a lowercase z is␣
 ↪122.
# This encoding is called ASCII, which stands for "American Standard Code for␣
 ↪Information Interchange."
'A' < 'a'
'A' > 'z'
'abc' < 'abd'
'abc' < 'abcd'
print()

'Jan' in '01 Jan 1838'
'Feb' in '01 Jan 1838'
```

[ ]: False

[ ]:
```
date = input('Enter a date in the DD MTH YYYY: ')
'Jan' in date
```

[ ]: False

[ ]:
```
'a' in 'abc'
'A' in 'abc'

'' in 'abc'
'' in ''
```

[ ]: True

1.2 Choosing Which Statements to Execute

[ ]:
```
if <<condition>> :
    <<block>>
```

[ ]:
```
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic")
```

```
6.0 is acidic
```

[ ]:
```
# If the condition is false, the statements in the block aren't executed.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic")
```

[ ]:
```
# If we don't indent the block, Python lets us knows.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
print(ph, "is acidic")
```

```
  Cell In[8], line 3
    print(ph, "is acidic")
    ^
IndentationError: expected an indented block
```

[ ]:
```
# Since we're using a block, we can have multiple statements that are executed␣
 ↪only if the condition is true.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic")
    print("You should be careful with that!")
```

```
6.0 is acidic
You should be careful with that!
```

[ ]:
```
# When we indent the first line of the block, the Python interpreter changes␣
 ↪its prompt to ... until the end of the block, which is signaled by a blank␣
 ↪line.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic")

print("You should be careful with that!")
```

```
You should be careful with that!
```

[ ]:
```
# If we don't indent the code that's in the block, the interpreter complains.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic.")

if ph > 7.0:
    print(ph, "is basic.")
```

[ ]:
```
# We can merge both cases by adding another condition/block pair using the elif␣
 ↪keyword (which stands for "else if"); each condition/block pair is called a␣
 ↪clause
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic.")
elif ph > 7.0:
    print(ph, "is basic.")
```

```
8.5 is basic.
```

[ ]:
```
# If the pH is exactly 7.0, neither clause matches, so nothing is printed.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    print(ph, "is acidic.")
elif ph > 7.0:
    print(ph, "is basic.")
```

[ ]:
```
# if the body of the first if changes the value of a variable used in the␣
 ↪second condition, they are not equivalent.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    ph = 8.0

if ph > 7.0:
    print(ph, "is basic.")
```

```
8.0 is basic.
```

[ ]:
```
# As a rule of thumb, if two conditions are related, use if/elif instead of two␣
 ↪ifs.
ph = float(input('Enter the pH level: '))
if ph < 7.0:
    ph = 8.0
elif ph > 7.0:
    print(ph, "is basic.")
```

[ ]:
```
# An if statement can be followed by multiple elif clauses.
compound = input('Enter the compound: ')
```

```
if compound == "H2O":
    print('Water')
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
```

```
Methane
```

[ ]:
```
# An if statement can be followed by multiple elif clauses.
compound = input('Enter the compound: ')
if compound == "H2O":
    print('Water')
elif compound == "NH3":
    print("Ammonia")
elif compound == "CH4":
    print("Methane")
else:
    print("Unknown compound")
```

```
Unknown compound
```

1.3 Nested if Statements

[ ]:
```
# An if statement inside another is called a nested if statement.
value = input('Enter the pH level: ')
if len(value) > 0:
    ph = float(value)
    if ph < 7.0:
        print(ph, "is acidic.")
    elif ph > 7.0:
        print(ph, "is basic.")
    else:
        print(ph, "is neutral.")
else:
    print("No pH value was given!")
```

```
No pH value was given!
```

1.4 Remembering Results of a Boolean Expression Evaluation

[ ]:
```
# Take a look at the following line of code and guess what value is assigned to␣
↪x.
x = 15 > 5 # Answer is True
```

Suppose you want to calculate someone's risk of heart disease using the following rules based on age and body mass index (BMI).

[ ]:
```
# Type 1 (nest if statement)
if age < 45 :
    if bmi < 22.0:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if bmi < 22.0:
        risk = 'medium'
    else:
        risk = 'high'
```

[ ]:
```
# Type 2 (Boolean expression)
young = age < 45
slim = bmi < 22.0
if young:
    if slim:
        risk = 'low'
    else:
        risk = 'medium'
else:
    if slim:
        risk = 'medium'
    else:
        risk = 'high'
```

[ ]:
```
# Type 3 (without nesting)
young = age < 45
slim = bmi < 22.0
if young and slim:
    risk = 'low'
elif young and not slim:
    risk = 'medium'
elif not young and slim:
    risk = 'medium'
elif not young and not slim:
    risk = 'high'
```

Reference * Title: Physics Programming Lecture Note (INU) * Author: Jeongwoo Kim, Ph.D. * Availability: https://sites.google.com/view/jeongwookim

---