Ch04-Conditionals

October 30, 2020

1 Conditionals - flow-control structures

- http://openbookproject.net/thinkcs/python/english3e/conditionals.html

1.1 Topics

- conditional statements and types
- comparision operators
- Truth tables

1.2 Conditional/Logical Structures

- conditional statements control the flow of execution of codes
  - makes some block of code to skip or execute based on some conditions
  - loosely speaking, it helps computer think and make decision
- boolean values - True or False
- boolean expressions - comparison operators are used to compare values that result in True or False

1.3 Comparison operators

- allow us to compare data values resulting in True or False outcome
- these are binary operators; take two operands where left hand side is compared with right hand side
- == (equal to)
- != (not equal to)
- > (greater than)
- >= (greater than or equal to)
- < (less than)
- <= (less than or equal to)
- comparison operators are used to create conditional statements

[2]:
```
# Examples of various comparison operators
x = 5
y = 10
```

[3]:
```
# Is x equal to y? True or False?
x == y
```

[3]: False

[4]:
```
# Is x not equal to y? True of False?
x != y
```

[4]: True

[5]:
```
# Is x is greater than y? True or False?
x > y
```

[5]: False

[6]:
```
# Is x less than y?
x < y
```

[6]: True

[7]:
```
# Is x less than or equal to y?
x <= y
```

[7]: True

[8]:
```
# Is x greater than or equal to y
x >= y
```

[8]: False

1.4 Conditional execution

- execute or skip a block of code when some condition is met
- conditional statments are created using keyword **if (condition)**
- three types of conditional selectors
- One-way, Two-way and Multi-way selectors

1.5 One-way selector

- just an if statement by itself
- syntax:

```
if <boolean expression> == True:
    # execute this block of code
```

- boolean expression can also comapre to False; the ultimate comparison result has to be True!

1.5.1 Visualize in PythonTutor.com

[10]:
```
if True:
    print('Hello, there!')
```

```
Hello, there!
```

[9]:
```
a = 'apple'
if a == 'apple': # this should evaluate True
    print('value of a equals to apple')
print('continue execution here...')
```

```
value of a equals to apple
continue execution here…
```

[12]:
```
a = 'apple'
if a == 'Apple': # this should evaluate False
    print('value of a equal to Apple') # this block will NOT be executed!
print('continue execution here...')
```

```
continue execution here…
```

1.6 Two-way selector

- if statement followed by else statement
- syntax:

```
if <boolean expression> == True:
    # execute this block of code
else:
    # otherwise, execute this
```

1.6.1 Visualize in PythonTutor.com

[13]:
```
num1 = 100.5
num2 = 100.49999999999999

if  num1 > num2:
    print('{} is greater than {}'.format(num1, num2))
else:
    print('{} is NOT greater than {}'.format(num1, num2))
```

```
100.5 is greater than 100.49999999999999
```

[14]:
```
if 10 >= 20:
    print ('print 10 is greater than or equal to 20')
else:
    print ('print 10 is NOT greater than or equal to 20')
```

```
print 10 is NOT greater than or equal to 20
```

1.7 Multi-way selector

- similar to multiple-choice questions with only one valid answer
- start from first **if statement**, if a **conditional expression** is evaluated True, the rest **elif** and **else** are ignored

```
if <condition1>:
    # block1
    # execute this...
elif <condition2>:
    # block2
    # execute this
...
...
else:
    # if all the previous conditions are evaluated False; this is the alternative!
    # else block
    # execute code in this block
```

[15]:
```
# Guess the number entered by user
ans = int(input('Enter a number between 1-5: '))
if ans == 1:
    print('You entered 1')
elif ans == 2:
    print('You entered 2')
elif ans == 3:
    print('You entered 3')
elif ans == 4:
    print('You entered 4')
elif ans == 5:
    print('You entered 5')
elif ans > 5:
    print('You entered a number larger than 5')
else:
    print('Hmm.... I do not know what you entered!')
```

```
Enter a number between 1-5: 2
You entered 2
```

1.7.1 Visualize in PythonTutor.com

1.8 Logical Operators

and, **or**, and **not** – allow to build more complex boolean expressions

Truth table for and

a	b	a **and** b
T	T	T
T	F	F
F	T	F
F	F	F

Truth table for or

a	b	a **or** b
T	T	T
T	F	T
F	T	T
F	F	F

Truth table for not

a	not a
T	F
F	T

1.8.1 Order of Evaluations of operators and expressions:

- Highest to Lowest: http://www.informit.com/articles/article.aspx?p=459269&seqNum=11

[19]:
```
# and demo
num = 10
if (num%2 == 0 and num > 0):
    print(f"{num} is even and positive")
```

```
10 is even and positive
```

[18]:
```
# and demo
num = -14
if num %2 == 0 and num < 0:
    print(f"{num} is even and negative")
```

```
-14 is even and negative
```

[20]:
```
# or demo
# a retimerment calculator
money = int(input('How much money have you saved? '))
ferrari = input('Do you have a farrari? [y/yes | n/no]: ')
if money >= 1000000 or ferrari == 'y' or ferrari == 'yes':
    print('Congrats!! You can retire now.')
else:
```

```
    print('Sorry, keep working!!')
```

```
How much money have you saved? 100000
Do you have a farrari? [y/yes | n/no]: yes
Congrats!! You can retire now.
```

[35]:
```
# not example
if not False:
    print('True')
```

```
True
```

[23]:
```
# not demo
# a retimerment calculator
money = int(input('How much money have you saved? '))
ferrari = input('Do you have a farrari? [y/yes | n/no]: ')
if not (money >= 1000000 or ferrari == 'y' or ferrari == 'yes'):
    print('Sorry, keep working!!')
else:
    print('Congrats!! You can retire now.')
```

```
How much money have you saved? 10
Do you have a farrari? [y/yes | n/no]: y
Congrats!! You can retire now.
```

1.9 Nested conditionals

- conditional statements can be nested inside another conditional statements
- syntax:

```
if condition:
    if condition1:
        # do something
    else:
        # do something else
        if condition2:
            # do something...
else:
    # do this...
```

[24]:
```
# program that determines if a given number is even or odd, positive or␣
 ↪negative
num = -100
if num == 0:
    print(f'{num} is zero!')
elif num%2 == 0:
    if num > 0:
        print(f"{num} is even and positive")
    else:
```

```
        print(f'{num} is even and negative')
else:
    if num > 0:
        print(f"{num} is odd and positive")
    else:
        print(f'{num} is odd and negative')

print('done')
```

```
-100 is even and negative
done
```

1.9.1 Visualize in PythonTutor.com

1.10 Exercises

Exercise 1: Write a program to test whether a given whole number is even, odd or zero.

[46]:
```
x = input('Enter a whole number: ')
x = int(x)
if x == 0:
    print(x, 'is zero')
elif x%2 == 0:
    print(x, 'is even')
else:
    print(x, 'is odd')
```

```
Enter a whole number:1
1 is odd
```

Exercise 1.1: Rewrite Exercise 1 using a function and write 2 test cases.

[6]:
```
def isEvenOddOrZero(num):
    if num == 0:
        return 'zero'
    elif num%2 == 0:
        return 'even'
    else:
        return 'odd'
```

[7]:
```
def test():
    assert isEvenOddOrZero(10) == 'even'
    assert isEvenOddOrZero(0) == 'zero'
    assert isEvenOddOrZero(19) == 'odd'
    print('all test cases passed...')
```

[8]:
```
test()
```

```
all test cases passed…
```

Exercise 2. Write a function that returns whether the given whole number is positive or negative. Also write at least 2 test cases.

[47]:
```
def positiveNegativeOrZero(num):
    if num == 0:
        return 'zero'
    elif num > 0:
        return 'positive'
    else:
        return 'negative'
```

[48]:
```
def test_positiveNegativeOrZero():
    assert positiveNegativeOrZero(0) == 'zero'
    assert positiveNegativeOrZero(100) == 'positive'
    assert positiveNegativeOrZero(-99.99) == 'negative'
    print('all test cases passed for positiveNegativeOrZero()')
```

[49]:
```
test_positiveNegativeOrZero()
```

```
all test cases passed for positiveNegativeOrZero()
```

Exercise 3: Write a program that converts students' grade value (0-100) to corresponding letter grade (A-F) where

`90 and above is A 80 and above is B 70 and above is C 60 and above is B 59 and below is F`

Write test cases to test each outcome.

Exercise 4: Write a program that helps someone decide where to go eat lunch depending on amount of money one has in their pocket.

Exercise 5: Given a day of week as integer (say 1 for Sunday) write a program that tells whether that day is weekend, or weekday and the actual name of the day.

Exercise 6: Write a program that determines whether someone is eligible to vote in the US federal election.

Exercise 7: Write a function day_name that converts an integer number 0 to 6 into the name of a day. Assume day 0 is "Sunday". Once again, return None if the argument to the function is not valid.

[1]:
```
def day_name(day):
    pass
```

[2]:
```
# Here are some tests that should pass
def test_day_name():
    assert day_name(3) == "Wednesday"
    assert day_name(6) == "Saturday"
    assert day_name(42) == None
    print('all test cases passed for day_name()')
```

[3]:
```
test_day_name()
```

```
---------------------------------------------------------------------------
AssertionError                            Traceback (most recent call last)
<ipython-input-3-97913494fc66> in <module>()
----> 1 test_day_name()

<ipython-input-2-3f7d21d757ff> in test_day_name()
      1 def test_day_name():
----> 2     assert day_name(3) == "Wednesday"
      3     assert day_name(6) == "Saturday"
      4     assert day_name(42) == None
      5     print('all test cases passed for day_name()')

AssertionError:
```

Exercise 8: Write a function that helps answer questions like ‘“Today is Wednesday. I leave on holiday in 19 days time. What day will that be?”’ So the function must take a day name and a delta argument (the number of days to add) and should return the resulting day name.s

[9]:
```
def day_add(dayName, delta):
    pass
```

[5]:
```
# Here are some tests that should pass
def test_day_add():
    assert day_add("Monday", 4) ==  "Friday"
    assert day_add("Tuesday", 0) == "Tuesday"
    assert day_add("Tuesday", 14) == "Tuesday"
    assert day_add("Sunday", 100) == "Tuesday"
    assert day_add("Sunday", -1) == "Saturday"
    assert day_add("Sunday", -7) == "Sunday"
    assert day_add("Tuesday", -100) == "Sunday"
    print('all test cases passed for day_add()')
```

[ ]:
```
test_day_add()
```

1.11 Kattis problems

- almost every problem involve some form of conditional statements

[ ]:
```

```