

ANL251 Python Programming

Study Unit 2

Control Flow and Lists

Learning Outcomes and Learning Resources

1. Compose appropriate Boolean expressions for given scenarios
 - SU2 Chapter 1.1
 - Textbook Videos and Exercises 27, 28
2. Construct conditional control flow
 - SU2 Chapter 1.2
 - Textbook Videos and Exercises 29 ~ 31
3. Use while-loop for repeated tasks
 - SU2 Chapter 1.3
 - Textbook Video and Exercise 33

4. Select lists as the appropriate data structures for given scenarios, and implement subsetting on Lists
 - SU2 Chapters 2.1, 2.2
 - Textbook Videos and Exercises 34, 38
 - <https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range>
 - <https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>
5. Solve problems using Python lists and for-loop
 - SU2 Chapter 2.3
 - Textbook Video and Exercise 32

Seminars: discussion and activities to reinforce students' understanding

1. Boolean Expressions

Recap

- Boolean expressions (SU2 Chapter 1.1, Textbook Videos and Exercises 27, 28)
- The Python type **bool** has two values: **True** and **False**.
- The *comparison operators* take two values and produce a Boolean value.
- There are also logical operators that produce Boolean values: **and**, **or**, and **not**.


Discussion

- What are the Python data types we have learned so far?
- What are the Python operators we have learned so far

Recap

Operators

Precedence(<https://docs.python.org/3/reference/expressions.html#operator-precedence>)



Low	<code>or</code>	Boolean OR
	<code>and</code>	Boolean AND
	<code>not x</code>	Boolean NOT
	<code>in, not in, is, is not, <, <=, >, >=, !=, ==</code>	Comparisons, including membership tests and identity tests
	<code> </code>	Bitwise OR
	<code>^</code>	Bitwise XOR
	<code>&</code>	Bitwise AND
	<code><<, >></code>	Shifts
	<code>+, -</code>	Addition and subtraction
	<code>*, @, /, //, %</code>	Multiplication, matrix multiplication, division, floor division, remainder [5]
High	<code>+x, -x, ~x</code>	Positive, negative, bitwise NOT
	<code>**</code>	Exponentiation [6]

Quiz

Expression	Result
not 80 >= 50 or 90 >= 50	
not ((80 >= 50) or (90 >= 50))	
not (80 >= 50)	
(80 >= 50) and (70 <= 50)	
(80 >= 50) or (70 <= 50)	
50 >= 50 and 85 >= 50	

Discussion

The minimum passing grade is 50. Variable grade refers to the grade for a student. Do the expressions below correctly represent the English sentence: "The student passed."?

`grade >= 50`

`not (grade < 50)`

`50 > grade`

`not not (grade >= 50)`

Discussion

The minimum passing grade is 50. The variables `math_grade`, `bio_grade`, and `cs_grade` represent a student's final grades in three courses. Write the boolean expressions to correctly represent each of the English sentences below:

The student passed none of the courses

The student passed at least one of the courses.

The student passed all of the courses.

The student passed some but not all of the courses

Discussion

The minimum passing grade is 50, the minimum grade for "A" is 80 and variables `math_grade`, `bio_grade`, and `cs_grade` represent a student's final grades in three courses. Write the expression that represents the English sentence:

The student passed all their courses and earned at least one A.

2. Conditional Statements

Recap

Conditional statements (SU2 Chapter 1.2, Textbook Videos and Exercises 29 ~ 31)

```
people = 30
cars = 40
trucks = 15

if cars > people:
    print("We should take the cars.")
elif cars < people:
    print("We should not take the cars.")
else:
    print("We can't decide.")

if people > trucks:
    print("Alright, let's just take the trucks.")
else:
    print("Fine, let's stay home then.")
```

Figure 2.1 Creating sequential if-elif-else blocks

Note: Each indentation level should be indented by **4 spaces**. As Python requires indentation to be consistent, it is important not to mix tabs and spaces. You should never use tabs for indentation. Instead, all indentation levels should be 4 spaces.

Discussion

Conditional statements

1. Change the numbers of cars, people, and trucks, and then trace through each if-statement to see what will be printed.
2. Try some more complex Boolean expressions like `cars > people` or `trucks < cars`.

Quiz

What is printed when the code below is executed?

```
def howbig(n):  
    if n > 100:  
        return "It's huge."  
    elif n > 10:  
        return "It's pretty big."  
    else:  
        return "It's not so big."  
  
print(howbig(150))
```

Quiz

Correct the syntax error for each of the code below if there is any.

```
if 18 < temp and temp < 26:  
    print("A comfortable temperature")  
else:  
    print("Be prepared for anything")  
elif temp < 18:  
    print("Better bring a sweater")
```

```
if temp < 0:  
    print("It's toque weather!")
```

Discussion

Simplify the code below.

```
if temp > 28:  
    print(True)  
elif temp < 12:  
    print(True)  
else:  
    print(False)
```

Discussion

Simplify the code below.

```
if temperature > 28:  
    if money > 0.99:  
        print("I'm buying a lemonade.")
```

Quiz

Variables `grade1` and `grade2` represent grades in two courses. Write the code to count the number of courses passed (with a 50 or higher).

Nested conditional statements

```
print("""You enter a dark room with two doors.
Do you go through door #1 or door #2?""")
door = input("> ")

if door == "1":
    print("There's a giant bear here eating a cheese cake.")
    print("What do you do?")
    print("1. Take the cake.")
    print("2. Scream at the bear.")
    bear = input("> ")
    if bear == "1":
        print("The bear eats your face off. Good job!")
    elif bear == "2":
        print("The bear eats your legs off. Good job!")
    else:
        print(f"Well, doing {bear} is probably better.")
        print("Bear runs away.")
elif door == "2":
    print("You stare into the endless abyss at Cthulhu's retina.")
    print("1. Blueberries.")
    print("2. Yellow jacket clothespins.")
    print("3. Understanding revolvers yelling melodies.")
    insanity = input("> ")
    if insanity == "1" or insanity == "2":
        print("Your body survives powered by a mind of jello.")
        print("Good job!")
    else:
        print("The insanity rots your eyes into a pool of muck.")
        print("Good job!")
else:
    print("You stumble around and fall on a knife and die. Good job!")
```

Figure 2.2 Creating nested if-elif-else blocks

3. while-loop

Recap

while-loop for repeated tasks (SU2 Chapter 1.3, Textbook Video and Exercise 33)

```
i = 0
print(f"At the top i is {i}")
while i < 6:
    i = i + 1
    print("i now is: ", i)
```

Figure 2.3 Creating a while-loop block

Note: while-loop is a type of unbounded loop, i.e. it does not stop if we set an always True condition.

Quiz

What will be printed when executing the code below?

```
num = 6
while num > 0:
    num = num - 2
    print(num)
```

Discussion

Write a program to ask the user for a "yes" or "no" input and continue asking until the user gives a valid response. Print the answer.

Discussion

validating user inputs: The program expects numeric inputs from the user for the two variables height, weight. What if the user inputs non-numeric characters?

```
name = input("Name? ")
age = input("How old are you? ")
height = input("How tall are you in metres? ")
weight = input("How much do you weigh in kilograms? ")

print(f"{name} is {age} old, {height} tall and {weight} heavy.")

bmi = float(weight)/float(height)**2
print(f"Your BMI is {bmi}.")
```

Study [Handling Exceptions](#) and improve the code above.

4. Operations on lists

Recap

Operations on lists (SU2 Chapters 2.1, 2.2, Textbook Videos and Exercises 34, 38)

```
[>>> animals = ['bear', 'tiger', 'penguin', 'zebra']
[>>> animals[1]
'tiger'
[>>> animals[1:]
['tiger', 'penguin', 'zebra']
[>>> animals[1:2]
['tiger']
[>>> animals[:2]
['bear', 'tiger']
[>>> animals[-1]
'zebra'
[>>> animals[-2]
'penguin'
[>>> animals[::2]
['bear', 'penguin']
```

Figure 2.4 Creating and subsetting Python list

Discussion

Common sequence operations on lists (also applicable to **tuple** in SU3)

<https://docs.python.org/3/library/stdtypes.html#sequence-types-list-tuple-range>

Operation	Result
<code>x in s</code>	<code>True</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>False</code>
<code>x not in s</code>	<code>False</code> if an item of <code>s</code> is equal to <code>x</code> , else <code>True</code>
<code>s + t</code>	the concatenation of <code>s</code> and <code>t</code>
<code>s * n</code> or <code>n * s</code>	equivalent to adding <code>s</code> to itself <code>n</code> times
<code>s[i]</code>	<code>i</code> th item of <code>s</code> , origin 0
<code>s[i:j]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code>
<code>s[i:j:k]</code>	slice of <code>s</code> from <code>i</code> to <code>j</code> with step <code>k</code>
<code>len(s)</code>	length of <code>s</code>
<code>min(s)</code>	smallest item of <code>s</code>
<code>max(s)</code>	largest item of <code>s</code>
<code>s.index(x[, i[, j]])</code>	index of the first occurrence of <code>x</code> in <code>s</code> (at or after index <code>i</code> and before index <code>j</code>)
<code>s.count(x)</code>	total number of occurrences of <code>x</code> in <code>s</code>

Quiz

If the variable `grades` refers to `[80, 90, 70, 24]`, what does each of the following evaluate to?

`grades[1]`

`grades[1:2]`

`grade[-1]`

`grade[3:]`

`len(grades)`

`min(grades)`

`max(grades)`

`sum(grades)`

Recap

Operations on lists

```
stuff = ['Apples', 'Oranges', 'Crows', 'Telephone', 'Light', 'Sugar']
print(stuff)

more_stuff = ["Day", "Night", "Song", "Frisbee", "Corn", "Banana", "Girl", "Boy"]
while len(stuff) != 10:
    next_one = more_stuff.pop()
    print("Adding: ", next_one)
    stuff.append(next_one)
    print(f"There are {len(stuff)} items now.", stuff)

print("We are removing the last element", stuff.pop())
print("We are joining all the elements", ' '.join(stuff))
print("We are joining the 4th and 5th elements with a #" , '#'.join(stuff[3:5]))
```

Figure 2.5 Adding, removing and joining elements in Python list

Discussion

Operations on lists

(<https://docs.python.org/3/library/stdtypes.html#mutable-sequence-types>)

NOT
applicable
to **tuple** in
SU3

Operation	Result
<code>s[i] = x</code>	item <i>i</i> of <i>s</i> is replaced by <i>x</i>
<code>s[i:j] = t</code>	slice of <i>s</i> from <i>i</i> to <i>j</i> is replaced by the contents of the iterable <i>t</i>
<code>del s[i:j]</code>	same as <code>s[i:j] = []</code>
<code>s[i:j:k] = t</code>	the elements of <code>s[i:j:k]</code> are replaced by those of <i>t</i>
<code>del s[i:j:k]</code>	removes the elements of <code>s[i:j:k]</code> from the list
<code>s.append(x)</code>	appends <i>x</i> to the end of the sequence (same as <code>s[len(s):len(s)] = [x]</code>)
<code>s.clear()</code>	removes all items from <i>s</i> (same as <code>del s[:]</code>)
<code>s.copy()</code>	creates a shallow copy of <i>s</i> (same as <code>s[:]</code>)
<code>s.extend(t)</code> or <code>s += t</code>	extends <i>s</i> with the contents of <i>t</i> (for the most part the same as <code>s[len(s):len(s)] = t</code>)
<code>s *= n</code>	updates <i>s</i> with its contents repeated <i>n</i> times
<code>s.insert(i, x)</code>	inserts <i>x</i> into <i>s</i> at the index given by <i>i</i> (same as <code>s[i:i] = [x]</code>)
<code>s.pop([i])</code>	retrieves the item at <i>i</i> and also removes it from <i>s</i>
<code>s.remove(x)</code>	remove the first item from <i>s</i> where <code>s[i]</code> is equal to <i>x</i>
<code>s.reverse()</code>	reverses the items of <i>s</i> in place

Quiz

If the variable `grades` refers to `[80, 90, 70, 24]`, what does each of the following evaluate to?

`grades[1]`

`grades[1:2]`

`grade[-1]`

`grade[3:]`

`len(grades)`

`min(grades)`

`max(grades)`

`sum(grades)`

Quiz

What will `grades` refer to after this code is executed?

```
grades = [80, 70, 60, 90]  
grades.sort()  
grades.insert(1, 95)
```

Discussion

Write a program to add each user input into a list until the user types enter to end.

4. Lists and for-loop

Recap

lists and for-loop (SU2 Chapter 2.3, Textbook Video and Exercise 32)

```
the_count = [1, 2, 3, 4, 5]
fruits = ['apples', 'oranges', 'pears', 'apricots']

for number in the_count:
    print(f"This is count {number}")

for fruit in fruits:
    print(f"A fruit of type: {fruit}")
```

Figure 2.6 Iterating elements in Python list using for-loop

Discussion

What's the first line to be printed after executing the code below?
What will be printed if s is initialized as an empty string?

```
s = 'good day'  
for char in s:  
    print(char)
```


Discussion

for-loop over indices:

There are problems where knowing the value of the items in a list or the characters in a string is not enough; we need to know where it occurs (i.e. its index).

Question 1: Write a program to shift each item in a list one position to the left and shift the first item to the last position.

Question 2: Write a program to count the corresponding characters of the two strings that are the same character. Your program may assume the two strings have the same length.

Choosing Test Cases

Discussion

```
#count the number of vowels (a, e, i, o, and u) in the user input.
s = input("This program counts the number of vowels in your input: ")
num_vowels = 0

for char in s:
    if char in 'aeiou':
        num_vowels = num_vowels + 1

print(f"The total number of vowels in your input is {num_vowels}.")
```

Which test cases should you choose in order to thoroughly test your code?

It is not realistic to test using every single possible input. Instead, we create relevant categories, and choose one representative **from each category**.

Discussion: Anymore test cases to add? Can the code above pass all the test cases you choose?

Input	Expected Output
"	0
'a'	1
'b'	0
'pffft'	0
'bandit'	2
'aeioua'	6

Choose test cases to thoroughly test the code in Figure 2.2

```
print("""You enter a dark room with two doors.
Do you go through door #1 or door #2?""")
door = input("> ")

if door == "1":
    print("There's a giant bear here eating a cheese cake.")
    print("What do you do?")
    print("1. Take the cake.")
    print("2. Scream at the bear.")
    bear = input("> ")
    if bear == "1":
        print("The bear eats your face off. Good job!")
    elif bear == "2":
        print("The bear eats your legs off. Good job!")
    else:
        print(f"Well, doing {bear} is probably better.")
        print("Bear runs away.")
elif door == "2":
    print("You stare into the endless abyss at Cthulhu's retina.")
    print("1. Blueberries.")
    print("2. Yellow jacket clothespins.")
    print("3. Understanding revolvers yelling melodies.")
    insanity = input("> ")
    if insanity == "1" or insanity == "2":
        print("Your body survives powered by a mind of jello.")
        print("Good job!")
    else:
        print("The insanity rots your eyes into a pool of muck.")
        print("Good job!")
else:
    print("You stumble around and fall on a knife and die. Good job!")
```

Figure 2.2 Creating nested if-elif-else blocks

*Thank
you*

