Python Control Structures

Computing for Data Analytics (CPSC 4800)

Mourad Bouguerra
mbouguerra@langara.ca

Langara College

June, 2022

Lesson's Outline

1. **Lesson's Learning Objectives**
2. **Set Data Structure**
3. **Python Control Structures**
   - Sequence
   - Selection
   - Iteration
     - for Loop
     - while Loop
     - Python range() Method
     - Python Random Module
     - List & Dictionary Comprehension

Learning Objectives

Learning Objectives

✎ Upon completion of this lesson, you will learn:

- ❒ What are the three control structures of any program?
  - ➡ Sequence control structures
  - ➡ Selection control structures
  - ➡ Iteration control structures
- ❒ How does Python implement the three control structures?

Python Tuple Data Structure

Sets

- A set is simlar to a list stores
  - a collection of data that are
    - immutable (unchangeable)
    - unordered
    - with duplicates
- A set can hold different data types
- To get the number of elements use the method *len*()

```
course_info = ('Computing for Data Analytics', 'CPSC 4800',3, \
                True,False,32.40)
print(f'The tuple {course_info} has {len(course_info)} elements.')
```

Python Set Data Structure

Sets

```
# empty set
set_0 = set()
# set with one element
set_1 = {'set with one item'}
# set from string
set_2 = set('Computing for Data Analytics')
# set from a list
set_3 = set(['List','Tuple','Dictionary','Set','List'])
print(f'{set_0}\n{set_1}\n{set_2}\n{set_3}\n')
```

Set Elements

- A set cannot have a mutable elements

```

set_1 = {1,2,3,('x','y')}
set_2 = {4800,[1,2,'CPSC']}
# Output
# TypeError: unhashable type: 'list'
```

Algorithm

- The terms
  - ✔ algorithm
  - ✔ effective procedure
  - ✔ mechanical procedure
  - ✔ software
  - ✔ computer program
  - ✔ computation
- `Step-by-step, well-defined, and finite instructions for solving a particular problem`

Algorithm

❒ **Böhm** and **Jacopini** proved mathematically in 1966 [1]

➡ any algorithm can be constructed using only three control structures

✔ Sequence ✔ Selection ✔ Iteration

Class Activity

❒ What is the output of the following Python program?

```
s = 'Sorting Order'
scores = [51, 54, 84, 30, 55, 51, 41, 86, 47, 59]
ruler = '-'*(len(s)+4)
print(f'\t\t{ruler}')
print(f'\t\t| Sorting Order |')
print(f'\t\t{ruler}')
print(f'\t 1 -- Ascending')
print(f'\t 2 -- Descending')
print(f'\t--|Scores\n\t--|{scores}')
order = int(input(f'\t--|Select you sorting order (1 or 2): '))
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Sequence

- A Python script will be executed in sequence
  - From left to right
  - From top to bottom
- Selection or iteration breaks the script execution sequence

Selection

- Selection, decision, or conditional control structure
  - breaks the program execution sequence
  - executes statements based on condition expression
- Python provides two selection constructs:
  - The if statements
  - The match case statements[a]

[a] Added in Python 10.

Selection

✎ if provides 3 types of tests:

- ➡ One-Way Test
  - ✔ if condition & body
- ➡ Two-Way Test
  - ✔ if condition & body
  - ✔ else body
- ➡ Mulit-Way Test
  - ✔ if condition & body
  - ✔ One or more elif condition & body
  - ✔ else body

One-Way Test

```
scores = [51, 54, 84, 30, 55, 51, 41, 86, 47, 59]
order = int(input(f'\t--|Select you sorting order (1 or 2): '))
scores.sort()
if order == 2:
    scores = scores[-1::-1]
print(f'\t--|Sorted scores\n\t--|{scores}')
```

Two-Way Test

```
scores = [51, 54, 84, 30, 55, 51, 41, 86, 47, 59]
order = int(input(f'\t--|Select you sorting order (1 or 2): '))
if order == 1:
    scores.sort()
else:
    scores.sort(reverse=True)
print(f'\t--|Sorted scores\n\t--|{scores}')
```

Multi-Way Test

```
scores = [51, 54, 84, 30, 55, 51, 41, 86, 47, 59]
order = int(input(f'\t--|Select you sorting order (1 or 2): '))
if order == 1:
    scores.sort()
elif order == 2:
    scores.sort(reverse=True)
else:
    print(f'You should enter only 1 or 2')
    print(f'Terminating script!')
    exit(1)
print(f'\t--|Sorted scores\n\t--|{scores}')
```

Class Activity

Selection Control Structures

- Write a Python script that will extract the time from date, and use a multi-way test to display a greeting according to the time of the day

  ① Good Morning

  ② Good Afternoon

  ③ Good Evening

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Python Selection Control Structure

match. . . case

```
s = 'Sorting Order'
ruler = '-'*(len(s)+4)
print(f'\t\t{ruler}')
print(f'\t\t| Sorting Order |')
print(f'\t\t{ruler}')
print(f'\t 1 -- Ascending')
print(f'\t 2 -- Descending')
print(f'\t 3 -- Quit')
choice = input(f'\t--|Select an option (1, 2 or 3): ')
match choice:
    case 1:
        print(f'\t--|You chose option 1')
    case 2:
        print(f'\t--|You chose option 2')
    case 3:
        print(f'\t--|You chose to quit!')
        exit(0)
```

Class Activity

Selection Control Structures

- Write a script that will display a menu of 5 commands and use a match cas to display the selected option

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Iteration

- Iteration, repetition, or loop control structures
  - allow you to execute a statement or a block of statements repetitively
  - Repetition is controlled by testing some boolean expression
- Python provides two iteration constructs:
  - The for loop
  - The while loop

Iteration Control Structures

For Loop

- For loop iterates over a sequence and mapping data types
  - str
  - tuple
  - list
  - dict

String for Loop

```
s = 'Data'
for c in s:
    print(c)
# Output
# D
# a
# t
# a
```

String for Loop

```
s = 'Data'
for i,c in enumerate(s):
    print(f'for i = {i}, c = {c}')
# Output
# for i = 0, c = D
# for i = 1, c = a
# for i = 2, c = t
# for i = 3, c = a
```

List for Loop

```
primes = [2, 3, 5, 7]
for p in primes:
    print(p)
# Output
# 2
# 3
# 5
# 7
```

List for Loop

```
primes = [2, 3, 5, 7]
for i, p in enumerate(primes):
    print(f'for i = {i}, p = {p}')
# Output
# for i = 0, p = 2
# for i = 1, p = 3
# for i = 2, p = 5
# for i = 3, p = 7
```

Tuple for Loop

```
primes = (2, 3, 5, 7)
for p in primes:
    print(p)
# Output
# 2
# 3
# 5
# 7
```

Tuple for Loop

```
primes = (2, 3, 5, 7)
for i, p in enumerate(primes):
    print(f'for i = {i}, p = {p}')
# Output
# for i = 0, p = 2
# for i = 1, p = 3
# for i = 2, p = 5
# for i = 3, p = 7
```

Dictionary for Loop

```
course_info = {'title':'Computing for Data Analytics',
               'code':'CPSC 4800','credit':3, \
                'is_transferable':True,'fees':32.40}
for x in course_info:
    print(x)
# Output
# title
# code
# credit
# is_transferable
# fees
```

Dictionary for Loop

```
course_info = {'title':'Computing for Data Analytics',
               'code':'CPSC 4800','credit':3, \
                'is_transferable':True,'fees':32.40}
for item in course_info.items():
    print(item)
# Output
# ('title', 'Computing for Data Analytics')
# ('code', 'CPSC 4800')
# ('credit', 3)
# ('is_transferable', True)
# ('fees', 32.4)
```

Dictionary for Loop

```
course_info = {'title':'Computing for Data Analytics',
               'code':'CPSC 4800','credit':3, \
                'is_transferable':True,'fees':32.40}
for key,item in course_info.items():
    print(f'(key,item)=({key},{item})')
# Output
# (key,item)=(title,Computing for Data Analytics)
# (key,item)=(code,CPSC 4800)
# (key,item)=(credit,3)
# (key,item)=(is_transferable,True)
# (key,item)=(fees,32.4)
```

while Loop

- while loop is used to iterate over a block statement
  - as long as the while condition is true

```
while condition:
    statement_1
    statement_2
    .
    .
    statement_n
```

while Loop

- Infinite while loop must be avoided by
  - ensuring the while condition will be become False
  - existing the loop after specific number iterations

```
while True:
    statement_1
    statement_2
    .
    .
    statement_n
```

Iteration Control Structures

while Loop

```
a = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
i = 0
while i < len(a):
    print(a[i], end=' ')
    i += 1
```

Iteration Control Structures

while Loop

```
a = [1, 2, 3, 4, 5, 6, 7, 8, 9]
i = 0
while i < len(a):
    print(a[i], end=' ')
    if (i+1) % 3 == 0:
        print()
    i += 1
```

while Loop

- break statement stops the execution of the loop
  - exits loop

```
i = 0
while True:
    print(f'*', end=' ')
    if (i+1) % 3 == 0:
        print()
    i += 1
    if i == 99:
        break
```

while Loop

- continue statement stops the execution of the current iteration
  - and goes to the next iteration

```
i = 0
while i < 10:
    i += 1
    if i == 4:
        continue
    print(f'{i} ', end=' ')
```

Class Activity

❒ What is the output of the following Python code?

```
i = 0
while i < 20:
    i += 1
    if i % 2 == 0:
        continue
    print(f'{i} ', end=' ')
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Class Activity

❒ What is the output of the following Python code?

```
i = 0
while True:
    print(f'*', end=' ')
    if (i+1) % 5 == 0:
        print()
    i += 1
    if i == 50:
        break
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Python range() Method

range() Method

- The Python range() method has the following syntax
  - ➡ range(start, end, step)
    - ✔ returns a sequence starting from start
    - ✔ increments by step
    - ✔ stops at end-1

```
for x in range(1, 10, 3):
    print(f'x = {x}')
# Output
# x = 1
# x = 4
# x = 7
```

Python range() Method

range() Method

- range(start, end, step)
  - If the start is not specified
    - ✔ The sequence starts from 0
  - If the step is not specified
    - ✔ The sequence increments by 1

```
for x in range(4):
    print(f'x = {x}')
# Output
# x = 0
# x = 1
# x = 2
# x = 3
```

Class Activity

- What is the output of the following Python code?

```
for x in range(10, 100, 20):
    print(f'x = {x}')
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- What is the output of the following Python code?

```
for x in range(100, 1, -30):
    print(f'x = {x}')
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- Use list comprehension to set up a list of all two-digit integers
- Use list comprehension to set up a list of all three-digit integers
- Use list comprehension to generate 25 pseudo random integers between 50 and 70 inclusive
- Use list comprehension to generate 35 pseudo random floating-points between 50.0 and 70.0 inclusive

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Python range() Method

random Module

- The Python random module provides
  - methods to generate pseudorandom numbers
    - ✔ randint(a,b) for random integers between a and b inclusive
    - ✔ random() for random floating-points in the interval $[0, 1)$
    - ✔ uniform(a,b) for random floating-points in the interval $[a, b]$

Class Activity

- Check out the help documentation for the following random module methods?
  - randint()
  - random()
  - uniform()

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

List Comprehension

Iteration

- We can use a for loop to populate a list

```
my_list = []
for x in range(0,11):
    my_list.append(x)
print(f'my_list = {my_list}')
```

- A list comprehension provides a shorter syntax to do the same

```
my_list = [x for x in range(0,11)]
print(f'my_list = {my_list}')
```

Class Activity

- What is the content of the following Python lists?

```
a = [x for x in range(10)]
b = [x for x in range(10, 100)]
c = [x for x in range(0, 30, 5)]
```

Chinese Proverb

I Hear & I Forget, I See & I Remember, I Do & I Understand

Class Activity

- What is the content of the following Python lists?

```python
a = [x for x in range(10) if x % 3 == 1]
b = [x for x in range(10, 100) if x % 4 == 0]
c = [x for x in range(0, 30, 5) if x % 10 == 0]
print(f'a = {a}')
print(f'b = {b}')
print(f'c = {c}')
```

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

Class Activity

- Use list comprehension to set up a list of all two-digit integers
- Use list comprehension to set up a list of all three-digit integers
- Use list comprehension to generate 25 pseudo random integers between 50 and 70 inclusive
- Use list comprehension to generate 35 pseudo random floating-points between 50.0 and 70.0 inclusive

Chinese Proverb

Tell Me & I Forget,
Teach Me & I Remember,
Involve Me & I Learn

C. Böhm and G. Jacopini.
Flow diagrams, turing machines and languages with only two formation rules.
Communications of the ACM, 9(366–371), 1966.