Basics of Python

Jin Wang

MSIS, Rutgers

Outline

Setup

1. Vairables and functions

2. Conditionals and loops

2.1 Conditionals

2.1 Loops

3. Strings and lists

- 3.1 Strings
- 3.2 list

4. Set, tuple, and dictionary

- 4.1 Set
- 4.2 Tuple
- 4.3 Dictionary

Setup

- Python
- vscode
  - codespace (online)
- github
  - My github: https://github.com/Devin-Wong
  - The course github repository: https://github.com/Devin-Wong/641ABI.git

1. Vairables and functions

String formatting

```
name = 'Jin'
print('Hello, ' + name)
```

```
n = 35
n_as_str = str(n)
print("The number of students is " + n_as_str)
```

f-string

```
x = 3
y = 4

z = x + y

print(f"{x} plus {y} equals {z}.")
```

Create our own function by `def`

First function without arguments

```
def hello():
        print("Hello, ")
```

Function with arguments

```
def hello(nm):
        print("Hello, ", end="")
        print(nm)
```

Function with default value

```python
# create hello() function
def hello(nm, greeting="Hello "):
        print(greeting, end="")
        print(nm)

name_1 = "Jin"
name_2 = "Mark"

# say "Hello" to Jin
hello(name_1)

# say "Hello" to Mark
hello(name_2)
```

Return values

```
def add(a, b):
        c = a + b
        return c
```

2. Conditionals and loops

2.1 Conditionals

Relational operators

- Python has a set of "**Operators**" that can be used to ask mathematical questions.

Symbol	meaning
> and <	larger and smaller
>=	greater than or equal to
<=	less than or equal to
==	equals
!=	not equal to

Logical operators

- `and` operator
- `or` operator

Conditional statement: : `if` statement

```python
x = float(input("Please input a number: "))

if x>0:
    print("It is a positive number.")
```

Alternative execution: Control Flow, `else`, and `elif`

```
D = float(input("Please input a number: "))

if x>0:
    print("It is a positive number.")
elif x==0:
    print("It is zero.")
else:
    print("It is a negative number.")
```

Pythonic `if`

- 'Pythonic' code means the code only seen in Python programming.

```
x = int(input("Please input an integer: "))

b = True if x!=0 else False

if b:
        print("x is not zero.")
else:
        print("x is zero")
```

2.1 Loops

`while` loops

```python
i = 3
while i>0:
    print("Hello")
    i-=1
```

- decrement, `i-=1`
- increment, `i+=1`

`for` loops

```python
for i in range(3):
    print("Hello")
```

`continue` and `break`

`break`

```python
while True:
    n = int(input("Please input a score (0-100): "))
    if 0<=n<=100:
        break

print(n)
```

continue

- Different from `break`, `continue` *tells Python to go to the next iteration for a loop and ignore the code after* `continue`. For example,

```
for i in range(4):
        if i == 1:
                continue

        print(i)
```

3. Strings and lists

3.1 Strings

```
>>> s = "Hello"
>>> letter = s[1]
>>> letter
```

Strings are immutable

`len` function

String slices

```
>>> s = "Hello, Jin!"
>>> s[2:5]
'llo'
>>> s[:2]
'He'
>>> s[-1]
'!'
>>> s[-2:]
'n!'
```

(Fruitful) String Methods

- `.strip()`
- `.title()`
- `.upper()`
- `.split()`

3.2 list

A list is a sequence

```
>>> list_1 = [10, 20, 30, 40]
>>> list_2 = ['Rutgers', 'Princeton', 'NYU']
```

`len()` function

List slices

Lists are mutable

(Void) List methods

- `.append()`
- `.sort()`

List operations

- `+` operator
- `*` operator
- `in` operator

Map

```
def square(x)
    return number ** 2

numbers = [1, 2, 3, 4, 5]

squared = map(square, numbers)

print(list(squared))
```

`lambda() function`

```
def divide(x, y):
    return x/y

# lambda function, which is equivalent to the above divide function.
divide = lambda x, y: x/y

print(divide(1,3))
```

-
  - Which is interesting in lambda functions is that *if we don't assign lambda function to a variable, Python will destroy it immediately.*

Lamdba function in `map()`

- Lambda function is often used in when a function is used only once.
- For example, in the above map program, we can use lambda function instead of defining an explicit function.

```
numbers = [1, 2, 3, 4, 5]

squared = map(lambda x: x**2, numbers)

print(list(squared))
```

`Filter`

```
numbers = [0, -1, 2, -3, 4]
positive_values = filter(lambda n: n>0, numbers)

print(list(positive_values))
```

Loop alternative: List comprehension

```
numbers = [1, 3, 5]

double_numbers = [n * 2 for n in numbers]

print(double_numbers)
```

Comprehensions with conditionals

```
numbers = [1, -2, 3, -4]
positive_numbers = [n for n in numbers if n > 0]
print(positive_numbers)
```

4. Set, tuple, and dictionary

4.1 Set

Defining sets

- Sets don't hold order
- Sets don't allow duplicate elements

```
>>> universities = {'Rutgers', 'NYU', 'Princeton'}
>>> universities
{'Rutgers', 'Princeton', 'NYU'}
```

list-like function and opertion

- `len()` function
- `in` operation

List and set are interchangeable

```
>>> l = [2, 0, 2, 3]
>>> s = set(l)
>>> s
{0, 2, 3}
>>> l_new = list(s)
[0, 2, 3]
```

(Fruitful) Set Methods and Operations

- `intersection()` method and `&` operator
- `union()` method and `|` operator
- `difference()` method and `-` operator

4.2 Tuple

Defining tuples

```
>>> t = 'a', 'b', 'c', 'd' # or t = ('a', 'b', 'c', 'd')
>>> t
('a', 'b', 'c', 'd')
```

list-like Slice operator

```
>>> t = ('a', 'b', 'c', 'd')
>>> t[0]
'a'
>>> t[1:3]
('b', 'c')
```

Tuples are immutable

Tuple assignment

```
a, b, c = 1, 2, 3
```

Tuples as return values

```
def get_quot_rem(x, y):
    return x//y, x%y

x, y = 7, 3
quot, rem = get_quot_rem(x, y)
```

`zip()` function

```
counties = ['MIDDLESEX', 'MIDDLESEX', 'SOMERSET']
cities = ['Piscataway', 'HIGHLAND PARK', 'FLAGTOWN']
zipcodes = ['08854', '08904', '08821']

print(zip(counties, cities, zipcodes))
```

`enumerate()` function

```
universities = ["Princeton", "MIT", "Harvard", "Stanford"]

for i, university in enumerate(universities):
    print(i+1, university)
```

4.3 Dictionary

- We can *define a dictionary* in the following way.

```
>>> eng2arabic = {'one': 1, 'two': 2, 'three': 3}
```

- We can *add new items*

```
>>> eng2arabic = {'one': 1, 'two': 2, 'three': 3}
>>> eng2arabic['four'] = 4
>>> eng2arabic
{'one': 1, 'two': 2, 'three': 3, 'four': 4}
```

- Define an empty dictionary, then add items.

```
>>> d = dict() # d would be an empty dictionary
>>> d['one'] = 1
>>> d['two'] = 2
>>> d
{'one': 1, 'two': 2}
```

loopup

- `len` function
- Lookup
- `in` operator (for keys, not values)

Dictionary and list

```
cities = ['PISCATAWAY', 'HIGHLAND PARK', 'FLAGTOWN']
zipcodes = ['08854', '08904', '08821']

d = dict(zip(cities, zipcodes))
print(d)

print(d['PISCATAWAY'])
```