Basics of Python

Jin Wang

MSIS, Rutgers

Outline

Setup

- Python
- vscode
  - codespace (online)
- github
  - My github: https://github.com/Devin-Wong
  - The course github repository: https://github.com/Devin-Wong/ABI-2025-Spring.git

1. Vairables and functions

1.1 Variables

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}.")
```

1.2 Functions

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

```
# 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
```

Practice Questions

1. Use `input` function to give show a prompt to
    - tell the user the program can return the product of two numbers, and
    - Let users input two numbers
    - At last, return the result.

    Following is the expected outcome.

```
You can input two numbers. I will return the product the two numbers.
Please input the first number: 2.3
Please input the second number: 3.4
2.3 times 3.4 equals 7.819999999999999.
```

2. Write a program to create a function `show_employee()` using the following conditions.
    - It should accept the employee's name and salary, and print both.
    - If the salary is missing in the function call, then assign default value 9000 to salary.

    Given

```
showEmployee("Ben", 12000)
showEmployee("Jack")
```

    Expected output:

```
Name: Ben, salary: 12000

Name: Jack, salary: 9000
```

3. Calculate the area of a circle. The mathematical formula is πr^2, where r denotes the radius, and π is the mathematical constant, 3.14.

    i. create a function named `square`, which returns the squared value given a number.

    ii. create a function named `area`, which returns the area of a circle given a radius. In this function, you need to call the `square` function defined in (1).

    iii. Create a main function to orgnize the program. In the main function, print a sentence like

    - "The area of a circle with radius 1 is 3.14." if you put $r = 1$, or
    - "The area of a circle with radius 2 is 12.56." if you put $r = 2$.

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

```
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")
```

2.2 Loops

`while` loops

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

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

`for` loops

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

`continue` and `break`

`break`

```
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)
```

Practice Questions

1. Create a function named `get_maximum` taking a list of numbers, which can return the maximum number in the list.

2. Create a function taking a list, which can print the even indices values, i.e., the 2nd value, 4th value, 6th value, ... For example, given a list `x = ['a', 'b', 'c', 'd']`, the output should print 'b' and 'd'. (try to use `continue` in the loop)

3. Calculate the sum of 1 to n.
    - Create a function named `get_number` which returns a positive number. In the function you use `input` function to let user input a positive integer. If user inputs a negative number, reprompt the user with "Please input a positive number: " .
    - Create a function named `get_sum` taking a number n and returning the sum of 1 to n.
    - Create a `main` function which calls the two functions above to calculate the sum of 1 to n.

4. Given a list of numbers, try to print the numbers starting with the first one and stop when the sum of the numbers collected is larger than 100.

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)
```

Practice Questions

1. Use two `input` function twice, one for inputing first name and another one for last name. Use `.strip()` and `.title()` methods to correct user's input. Then print "Last name, First name". For example, input: " jin" as first name, "wang" as last name, the output would be "Wang, Jin".

2. Given an address, e.g., "100 Rockafeller Road, Piscataway, NJ 08854". Try to split the string, and get the information, street, city, state, zipcode.

3. Find even numbers, using loop and `append()` method.
    - Method 1:
        - Define a list containing a list of numbers
        - Use `for` loop to traverse the elements in the list. If the element is an even number, put it into a new list using `append` method.
        - For example, given a list `[2, 3, 10, 17, 20]`, the result is [2, 10, 20].
    - Method 2:
        - Use comprehension to finish the question.

4. Find prime numbers
    - Create a function which takes an integer, return `True` if the number is a prime number and return `False` otherwise.
    - Define a list containing a list of numbers
    - Use `filter` function to obtain the prime numbers in the list and print them.
    - For example, given a list `[2, 3, 10, 17, 20]`, the prime numbers are 2, 3 and 17.

5. Capitalize all words in a list, using `map` function.
    - Create a function which takes a string and returns capitalized string.
    - Define a list containing several words/strings.
    - Use `map` function to capitalize all words in the list.
    - For example, given a list `['Rbs', 'Rutgers']`, the result is ['RBS', 'RUTGERS'].
    - Use comprehension to finish the question.

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'])
```

Practice questions

1. We've provided you with a list of lottery players, and also with 6 random lottery numbers. The random lottery numbers are generated like this:

```
import random
lottery_numbers = set(random.sample(list(range(22)), 6))
```

And the list of players we've given you are:

```
players = [
("Rolf", {1, 3, 5, 7, 11, 20}),
("Charlie", {2, 7, 9, 5, 12, 15}),
("Anna", {7, 8, 1, 3, 13, 16}),
("Jen", {4, 7, 3, 5, 12, 21})
]
```

Try to find out the number of winnings for each person. For example, if the lottery number is `6, 8, 9, 13, 16, 19`, you need to print:

```
Rolf won 0.
Charlie won 1.
Anna has won 3.
Jen has won 0.
```

2. Summary data. Create a function, named `summarize_data`, which takes a list of numbers and returns a tuple containing the minimum, mean, and the maximum values. For example, given `[1, 2, 5]`, it would return `1, 2.6666, 5`.
    - You may use some built-in functions, like `sum`, `min`, and `max`.

3. **(Optional)** Letter frequency. Create a function, named `letter_frequency`, which takes a string and prints the letters in the string (case-insensitive, or just use upper case) and corresponding frequency. For example,
    - Given 'Rutgers, RBS', the function would print `[('G', 1), ('U', 1), ('B', 1), ('T', 1), ('R', 3), (',', 1), ('E', 1), ('S', 2)]`

4. Character counts. Given a string, try to obtain the counts of the characters in it (case insensitive). For example, given "Rutgers, RBS", the output should be the dictionary, {'T': 1, 'R': 3, 'B': 1, 'E': 1, ' ': 1, 'S': 2, 'U': 1, 'G': 1}.

5. Reverse lookup. The following dictionary shown the rosters of the three courses. We can easily check the student names each course has. Now we would like to do reverse lookup. That is, we would like to get a dictionary showing the courses that each student is taking.

```
roster = {
        'STATS_385': ['Jack', 'Anne', 'John', 'Peter'],
        'PROG_388': ['Tom', 'Mark', 'Jack', 'Jin'],
        'LARGESCALE_487': ['Anne', 'John', 'Jin', 'Jack']
}
```

End