Python Programming Part 1

Instructor: Vision Wang

Email: xinwang35314@gmail.com

Part 1

- Numbers/Strings/Lists
- Control flow
- Input and Output

Numbers: int, float, complex

- **Int**, or integer, is a whole number, positive or negative, without decimals, of unlimited length.

```
>>> 1
1
>>> 3565622554887711
3565622554887711
>>> z = -325522
>>> print(z)
-325522
```

- **Complex** numbers are written with a “j” as the imaginary part.

```
>>> 3+5j
(3+5j)
>>> x = 1-5j
>>> print(x)
(1-5j)
>>> -6j
(-0-6j)
```

- **Float**, or “floating point number” is a number, positive or negative, containing one or more decimals.
- Float can also be scientific numbers with an “e” to indicate the power of 10.

```
>>> 35e3
35000.0
>>> 12E4
120000.0
>>> z = -87.7e3
>>> print(z)
-87700.0
```

Type Conversion

- You can convert from one type to another with the **int()**, **float()**, and **complex()** methods.

```
>>> x = 1
>>> a = float(x)
>>> print(a)
1.0
>>> print(type(a))
<class 'float'>
```

```
>>> y = 2.8
>>> b = int(y)
>>> print(b)
2
>>> print(type(b))
<class 'int'>
```

```
>>> x = 1
>>> c = complex(x)
>>> print(c)
(1+0j)
>>> print(type(c))
<class 'complex'>
```

Python Operators

- Arithmetic operators are used with numeric values to perform common mathematical operations.

Operator	Name	Example
+	Addition	x + y
-	Subtraction	x – y
*	Multiplication	x * y
/	Division	x / y
%	Modulus	x % y
**	Exponentiation	x ** y
//	Floor division	x // y

```
>>> 21%5
1
>>> 21//5
4
```

```
>>> tax = 12.5 / 100
>>> price = 100.50
>>> tax * price
12.5625
>>> round(_, 2)
12.56
```

Python Operators

- Comparison operators are used to compare two values.

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

```
>>> 3==4
False
>>> 3>4
False
>>> 3!=4
True
```

String

- You can assign string to a variable.
- You can assign a multiline string to a variable by using three quotes.

```
>>> a = "Hello"
>>> a
'Hello‘

>>> b = """And me too!
Though I am more excited
than the others."""
>>> print(b)
And me too!
Though I am more excited
than the others.
```

Slicing

- You can return a range of characters by using the slice syntax.

```
>>> b = "Hello, Jason!"
>>> print(b[1:4])
ell
```

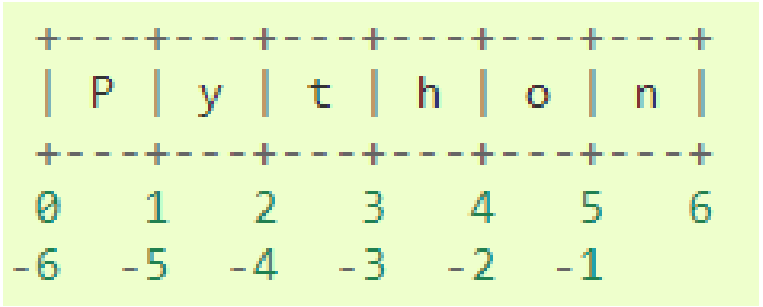
String Length

- To get the length of a string, use the **len()** function.

```
>>> print(len(b))
13
```

Negative Indexing

- Use negative indexes to start the slice from the end of the string.




```
>>> b = "Hello, Jason!"
>>> print(b[-5:-2])
aso
```

String Methods

- The **strip()** method removes whitespace from the beginning or the end

```
>>> a = "       Hello, Jason.   "
>>> print(a.strip())
Hello, Jason.
```

- The **lower()** method returns the string in lower case.
- The **upper()** method returns the string in lower case.

```
>>> b = "Hello, Jason."
>>> print(b.lower())
hello, jason.
>>> print(b.upper())
HELLO, JASON.
```

- The **replace()** method replaces a string with another string.

```
>>> m = "Because"
>>> print(m.replace("e","J"))
BJcausJ
```

- The **split()** method splicts the string into substrings if it find instances of the separator.

```
>>> m = "My favorite fruit is apple, banana, and orange."
>>> print(m.split(","))
['My favorite fruit is apple', ' banana', ' and orange.']
```

String Concatenation

```
>>> str1 = "Hello,"
>>> str2 = " Jason."
>>> print(str1+str2)
Hello, Jason.
```

```
>>> fruit = ", ".join(["Apple", "Banana",
"Pear"])
>>> print(fruit)
Apple, Banana, Pear
```

String Formatting

```
>>> day = 11
>>> month = "May"
>>> year = 2020
>>> text = "Today is {} {}, {}."
>>> print(text.format(month, day,
year))
Today is May 11, 2020.
```

```
>>> name = "John"
>>> age = 13
>>> print("%s is %d years old." %
(name, age))
John is 13 years old.
```

Lists

- Lists can be heterogeneous.
- Lists can be indexed and sliced.
- Lists can be manipulated.
- Return length using **len()** method.

```
>>> a = ["spam", "eggs", 100, 23, 2*3]
>>> a[-1]
6
>>> a[1]
'eggs'
>>> a[2] = a[2] +10
>>> print(a)
['spam', 'eggs', 110, 23, 6]
>>> print(a[0:3])
['spam', 'eggs', 110]
>>> print(len(a))
5
```

- Lists can be joined together.

```
>>> list1 = ["e","m","n"]
>>> list2 = [32,56,13]
>>> list3 = list1+list2
>>> print(list3)
['e', 'm', 'n', 32, 56, 13]
>>> list1.extend(list2)
>>> print(list1)
['e', 'm', 'n', 32, 56, 13]
```

Lists Methods

Method	Description
append()	Adds an element at the end of the list
clear()	Removes all the elements from the list
copy()	Returns a copy of the list
count()	Returns the number of elements with the specified value
extend()	Add the elements of a list (or any iterable), to the end of the current list
index()	Returns the index of the first element with the specified value
insert()	Adds an element at the specified position
pop()	Removes the element at the specified position
remove()	Removes the first item with the specified value
reverse()	Reverses the order of the list
sort()	Sorts the list

Control Flow:

if statement

```
x = 30
if x<= 15:
    y = x+15
elif x>=30:
    y = x+30
else:
    y = x
print("y =  ", y)
```

```
>>> y= 60
```

for loops

- The for loop is used to iterate over a sequence (list, tuple, string) .

```
>>> for x in [1,7,13,2]:
        print(x)

>>>
1
7
13
2
```

```
>>> genre = ['pop','rock','jazz']
>>> for i in range(len(genre)):
        print("I like ", genre[i])

>>>
I like  pop
I like  rock
I like  jazz
```

```
>>> for x in range(5):
        print(x)

>>>
0
1
2
3
4
```

range(n) generates a list of numbers [0,1,…,n-1]

while loops

- The while loop is used to iterate over a block of code as long as the test expression is true.
- Compared to for loop, we usually use while loop when we don't know the number of the times to iterate beforehand.

```
>>> x = 1
>>> while x<10:
        print(x)
        x = x+1

>>>
1
2
3
4
5
6
7
8
9
```

Loop Control Statements

- **break** – Jumps out of the closest enclosing loop.
- **continue** – Jumps to the top of the closest enclosing loop.
- **pass** – Does nothing, empty statement placeholder.

```
>>> for item in "string":
        if item =="i":
            break
        print(item)

>>>
s
t
r
```

```
>>> for item in "string":
        if item =="i":
            continue
        print(item)

>>>
s
t
r
n
g
```

Input

- Input using **input()** function

```
num = int(input("give me a number:"))
print(num)
print(type(num))
```

```
give me a number:3
3
<class 'int'>
```

- Input using **open()** function to open a file

File_object = open(“File_Name”, “Access_Mode”)

```
file1 = open(“test1.txt”, “r+”)
print(file1.readlines())
```

```
['eryiop']
```

Output

- We can use **format()** function to adjust output format.

Expressing a percentage:

```
>>> points = 20
>>> total = 22
>>> print("Correct answers:
{:.2%}".format(points/total))

Correct answers: 90.91%
```