Python Basics

Course 4- Coursera Week 1

Objectives

- Understand Python Types
- Understand Python Expressions and Variables
- Understand Python String Operations

What Is Python

Python is a general-purpose programming language that was created by Guido Van Rossum. Python is most praised for its elegant syntax and readable code. If you are just beginning your programming career Python suits you best. With Python you can do everything from **GUI development**, **Web application, System administration tasks, Financial calculation, Data Analysis, Visualization** and list goes on.

Some advantages:

- Free
- Powerful
- Widely used (Google, NASA, Yahoo, Electronic Arts, some UNIX scripts etc.)

Datatype & Variables

Variables are named locations that are used to store references to the object stored in memory. The names we choose for variables and functions are commonly known as Identifiers. In Python, Identifiers must obey the following rules.

1. All identifiers must start with a letter or underscore (_), you can't use digits. For e.g: `my_var` is a valid identifier but `1digit` is not.
2. Identifiers can contain letters, digits and underscores (_). For e.g: `error_404`, `_save` are valid identifiers but `$name$` (`$` is not allowed) and `#age` (`#` is not allowed) are not.
3. They can be of any length.
4. Identifiers can't be a keyword. Keywords are reserved words that Python uses for special purposes).

The following are keywords in Python 3

1	False	class	finally	is	return
2	None	continue	for	lambda	try
3	True	def	from	nonlocal	while
4	and	del	global	not	with
5	as	elif	if	or	yield
6	pass	else	import	assert	
7	break	except	in	raise	

Assigning Values to Variables

Values are basic things that programs work with. For e.g: 1, 11, 3.14, "hello" are all values.
In programming terminology, they are also commonly known as literals.
Literals can be of different types for e.g 1, 11 are of type int, 3.14 is a float and "hello" is a string.
Remember that in Python everything is object even basic data types like int, float, string. We will elaborate more on this in later chapters.

In Python, you **don't** need to declare types of variables ahead of time. The interpreter automatically detects the type of the variable by the data it contains. To assign value to a variable equal sign (=) is used. The = sign is also known as the assignment operator.

The following are some examples of variable declaration

```
x = 100                 # x is integer

pi = 3.14               # pi is float


empname = "python is great"   # empname is string


a = b = c = 100 # this statement indicates to assign 100 to c, b and a.
```

Comments

Comments are notes which describe the purpose of the program or how the program works. Comments are not programming statements that Python interpreter executes while running the program. Comments are also used to write program documentation. In Python, any line that begins with a pound sign (#) is considered a comment. For e.g:

```
# This program prints "hello world"

print("hello world")
```

We can also write comments at the end of a statement. For e.g:

```
# This program prints "hello world"

print("hello world")  # display "hello world"
```

Python Data Types

Python has 5 standard data types namely:

- Numbers
- Strings
- Lists
- Tuples
- Dictionaries
- Boolean - In Python, True and False are boolean literals. But the following values are also considered as false:
    - 0 - zero , 0.0
    - [] - empty list , () - empty tuple , {} - empty dictionary , ''
    - None

Python Numbers

This data type supports only numerical values like 1, 31.4, -1000, 0.000023, 88888888.

Python supports 3 different numerical types.

- int - for integer values like 1, 100, 2255, -999999, 0, 12345678.
- float - for floating-point values like 2.3, 3.14, 2.71, -11.0.
- complex - for complex numbers like 3+2j, -2+2.3j, 10j, 4.5+3.14j.

Determining types

Python has type() inbuilt function which is used to determine the type of the variable.

```
1   >>> x = 12
2   >>> type(x)
3    <class 'int'>
```

String Operations

Strings in python are contiguous series of characters delimited by single or double quotes.

Python **doesn't** have any separate data type for characters, so they are represented as a single character string.

Creating strings

You can use the following syntax to create strings.

```
1   >>> name = "tom" # a string
2   >>> mychar = 'a' # a character
```

You can **also** use the following syntax to create strings.

```
1   >>> name1 = str() # this will create an empty string object
2   >>> name2 = str("newstring") # string object containing 'newstring'
```

Operations on string

String index starts from 0, so to access the first character in the string type:

```
>>> name[0] #
```

```
1  name = "tom"
2  print(name[0])
3  print(name[1])
```

What would be the output?!

T

O

Operations on string # Cont..

The `+` operator is used to concatenate string and `*` operator is a repetition operator for string.

```
>>> s = "tom and " + "jerry"
>>> print(s)
tom and jerry
```

```
>>> s = "spamming is bad " * 3
>>> print(s)
'spamming is bad spamming is bad spamming is bad '
```

Slicing string

You can take subset of string from original string by using `[]` operator also known as slicing operator.

Syntax: `s[start:end.`

This will return part of the string starting from index `start` to index `end - 1.` Let's take some examples.

```
1   >>> s = "Welcome"
2   >>> s[1:3]
3    el
```

Note:
The `start` index and `end` index are optional. If omitted then the default value of `start` index is `0` and that of `end` is the last index of the string.

String Functions in Python

Function name	Function Description
len()	returns length of the string
max()	returns character having highest ASCII value
min()	returns character having lowest ASCII value

IN AND NOT IN OPERATORS

You can use **`in`** and **`not in`** operators to check the existence of a string in another string. They are also known as membership operator.

```
>>> s1 = "Welcome"
>>> "come" in s1
True
>>> "come" not in s1
False
```

Testing strings

String class in python has various inbuilt methods which allows to check for different types of strings.

Method Name	Method Description
isalnum()	Returns True if string is alphanumeric
isalpha()	Returns True if string contains only alphabets
isdigit()	Returns True if string contains only digits
isidentifier()	Return True is string is valid identifier
islower()	Returns True if string is in lowercase
isupper()	Returns True if string is in uppercase
isspace()	Returns True if string contains only whitespace

Searching for Substrings

Method Name	Method Description
endswith(s1: str): bool	Returns True if strings ends with substring s1
startswith(s1: str): bool	Returns True if strings starts with substring s1
count(substring): int	Returns number of occurrences of substring the string
find(s1): int	Returns lowest index from where s1 starts in the string, if string not found returns -1
rfind(s1): int	Returns highest index from where s1 starts in the string, if string not found returns -1

Converting Strings

Method Name	Method Description
capitalize(): str	Returns a copy of this string with only the first character capitalized.
lower(): str	Return string by converting every character to lowercase
upper(): str	Return string by converting every character to uppercase
title(): str	This function return string by capitalizing first letter of every word in the string
swapcase(): str	Return a string in which the lowercase letter is converted to uppercase and uppercase to lowercase
replace(old\, new): str	This function returns new string by replacing the occurrence of old string with new string