Lecture_1

January 7, 2023

1 LEC 1: Introduction to Python

1.1 Python Basics

[2]:
```
print("Hello, world!")
```

```
Hello, world!
```

[4]:
```
import math
print(math.pi)
```

```
3.141592653589793
```

1.1.1 Data Types

[2]:
```
b=-87
c=90.76
print(type(b))
print(type(c))
```

```
<class 'int'>
<class 'float'>
```

[3]:
```
d = 3 + 6j
print(d,type(d))
```

```
(3+6j) <class 'complex'>
```

[4]:
```
e = 0B1010 # binary
print(e, type(e))
```

```
10 <class 'int'>
```

[5]:
```
f= 0xff # hexadecimal
print(f, type(f))
```

```
255 <class 'int'>
```

[14]:
```
y = 0o456 # octal
print(y, type(y))
```

```
302 <class 'int'>
```

[2]:
```
g = True
print(type(g))
print(9>7)
```

```
<class 'bool'>
True
```

[7]:
```
x=33.5
h=int(x)
print(h)
```

```
33
```

[4]:
```
s = 'sagar' # string
print(type(s))
x =float('22.5')
print(x)
```

```
<class 'str'>
22.5
```

[3]:
```
print(bin(3)) # conversions
print(oct(10))
print(hex(16))
```

```
0b11
0o12
0x10
```

[11]:
```
p=0xab # covert from hexadecimal to octal
print(oct(p))
```

```
0o253
```

1.1.2 Variables

Python is strongly, dynamically typed.

[4]:
```
x = 1.0
print(x)

x = 2
print(x)
```

```
1.0
2
```

[6]:
```
y = "test"
```

[7]:
```
print(y)
```

```
test
```

[8]:
```
x = 1
x = "string"
print(x)
```

```
string
```

[9]:
```
x = 1
print(type(x))
```

```
<class 'int'>
```

[10]:
```
x = "string"
print(type(x))
```

```
<class 'str'>
```

[11]:
```
x = 0.1
print(type(x))
```

```
<class 'float'>
```

[12]:
```
x = 0.1
type(x)
```

[12]: `float`

1.1.3 Basic Arithmetic

Operators for integers: `+ - * / // % **`

Operators for floats: `+ - * / **`

Boolean expressions: * keywords: `True` and `False` (note capitalization) * `==` equals: `5 == 5` yields `True` * `!=` does not equal: `5 != 5` yields `False` * `>` greater than: `5 > 4` yields `True` * `>=` greater than or equal: `5 >= 5` yields `True` * Similarly, we have `<` and `<=`.

Logical operators: * `and`, `or`, and `not` * `True and False` * `True or False` * `not True`

[47]:
```
88/7  # 88 is dividend and 7 is divisor
```

[47]: `12.571428571428571`

[45]:
```
88//7  # gives quotient
```

[45]:
```
12
```

[46]:
```
88%7  # gives remainder
```

[46]:
```
4
```

[5]:
```
5**2
```

[5]:
```
25
```

1.1.4 Strings

Concatenation: `str1 + str2`

Printing: `print(str1)`

[13]:
```
str1 = "Hello, "
str2 = "World!"
str3 = str1 + str2
str3
```

[13]:
```
'Hello, World!'
```

[14]:
```
print(str3)
```

```
Hello, World!
```

[8]:
```
k = "you are awesome"
print(k)
print(k[0],k[1])
print(len(k))
```

```
you are awesome
y o
15
```

[9]:
```
# string slicing
print(k[0:6])
print(k[0:9:2])
print(k[::-1])
print(k[-1])
```

```
you ar
yuaea
emosewa era uoy
e
```

[29]:
```
print(k.find("a"))
print(k.count("a"))
```

```
4
2
```

Formatting:

[19]:
```
x = 23
y = 52
name = "Alice"

str1 = f"{name}'s numbers are {x} and {y}, and their sum is {x + y}"
str1
```

[19]:
```
"Alice's numbers are 23 and 52, and their sum is 75"
```

[6]:
```
str1 = "a: %s" % "sagar"

str1 = 'a:(1, hello)'

print(str1)

str2 = " %f, %s, %d " % (1.0, 'hello', 5)
print(str2)

str3 = "c: {} ".format(3.14)
print(str3)
```

```
a:(1, hello)
 1.000000, hello, 5
c: 3.14
```

[32]:
```
# some methods
str1 = "Hello, World!"
print(str1)
print(str1.upper())
print(str1.lower())
```

```
Hello, World!
HELLO, WORLD!
hello, world!
```

[33]:
```
str1.replace?
```

[34]:
```
str1.replace('l', 'p')
```

[34]:
```
'Heppo, Worpd!'
```

1.1.5 Control Flow

If statements:

[8]:
```
x = 2
y = 1
z = 3
if x == y:
    print("Hello") #indentation
elif x == z:
    print("Goodbye")
else:
    print("???")
```

```
???
```

For loops

[9]:
```
print("loop 1")
for i in range(6): # default - start at 0, increment by 1
    print(i)

print("\nloop 2")
for i in range(10, 2, -2): # inputs are start, stop, step
    print(i)
```

```
loop 1
0
1
2
3
4
5

loop 2
10
8
6
4
```

while loops

[1]:
```
i = 1
while i < 100:
    print(i**2)
    i += i**2  # i = i + i**2, a += b means a = a+b
```

```
1
4
```

```
36
1764
```

continue - skip the rest of a loop

break - exit from the loop

pass - does nothing

[12]:
```
for num in range(2, 10):
    if num % 2 == 0:
          pass    # this jumps us back to the top
    print(f"Found {num}, an odd number")
```

```
Found 2, an odd number
Found 3, an odd number
Found 4, an odd number
Found 5, an odd number
Found 6, an odd number
Found 7, an odd number
Found 8, an odd number
Found 9, an odd number
```

[8]:
```
n = 64
for x in range(2, n):
    if n % x == 0: # if n divisible by x
        print(f'{n} equals {x} * {n // x}')
        # break
```

```
64 equals 2 * 32
64 equals 4 * 16
64 equals 8 * 8
64 equals 16 * 4
64 equals 32 * 2
```

pass does nothing

[54]:
```
if False:
    pass # to implement
else:
    print('True!')
```

```
True!
```

1.1.6 Exceptions

[8]:
```
100 / 0
```

```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
```

```
~\AppData\Local\Temp/ipykernel_2116/3039867184.py in <module>
----> 1 100 / 0

ZeroDivisionError: division by zero
```

[6]:
```
try:
    x = 109 / 0
except ZeroDivisionError:
    print("We divided by zero")
```

```
We divided by zero
```

1.1.7 Functions

Functions are declared with the keyword `def`

[13]:
```
# def tells python you're trying to declare a function

def area(b, h):
    return 0.5 * b * h

area(3, 4)
```

[13]:
```
6.0
```

[12]:
```
def triangle_area(base, height):
    if base < 0 or height < 0:
        raise ValueError("Base and height must be non-negative")
    return 0.5 * base * height

triangle_area(2, -2)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_2116/2119930096.py in <module>
      4     return 0.5 * base * height
      5
----> 6 triangle_area(2, -2)

~\AppData\Local\Temp/ipykernel_2116/2119930096.py in triangle_area(base, height
      1 def triangle_area(base, height):
      2     if base < 0 or height < 0:
----> 3         raise ValueError("Base and height must be non-negative")
      4     return 0.5 * base * height
      5

ValueError: Base and height must be non-negative
```

[13]:
```
# everything in python is an object, and can be passed into a function
def f(x):
    return x+2

def twice(f, x):
    return f(f(x))

twice(f, 2) # + 4
```

[13]: 6

[15]:
```
def g(a, x, b=2):
    return a * x + b
```

[18]:
```
g(2, 5, 2)
```

[18]: 12

[17]:
```
g(2, 5)
```

[17]: 10

1.1.8 Exercise 1

1. Print every power of 2 less than 10,000
2. Write a function that takes two inputs, a and b and returns the value of $a + 2b$
3. Write a function takes a number n as input, and prints all Fibonacci numbers less than n

[33]:
```
# ex 3

n = int(input("Choose n: "))
a1 = int(input("Choose a1: "))
a2 = int(input("Choose a2: "))
print(a1)
print(a2)
a = a1 + a2

while a <= n:
    print(a)
    a1 = a2
    a2 = a
    a = a1 + a2
```

```
Choose n: 100
Choose a1: 0
Choose a2: 1
0
1
```

```
1
2
3
5
8
13
21
34
55
89
```

1.1.9 Lists

A list in Python is an ordered collection of objects

[6]:
```
a = ['x', 1, 3.5]
print(a)
```

```
['x', 1, 3.5]
```

Python indexing starts at 0.

[8]:
```
a.remove?
```

[9]:
```
a.remove(1)
print(a)
```

```
['x', 3.5]
```

You can append to lists using `.append()`, and do other operations, such as `pop()`, `insert()`, etc.

[1]:
```
a = []
for i in range(10):
    a.append(i**2)

print(a)
a.pop()
print(a)

a.insert(0,7)
a
```

```
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]
[0, 1, 4, 9, 16, 25, 36, 49, 64]
```

[1]:
```
[7, 0, 1, 4, 9, 16, 25, 36, 49, 64]
```

[13]:
```
while len(a) > 0:
    elt = a.pop()
    print(elt)
```

```
81
64
49
36
25
16
9
4
1
0
```

Python terminology: * a list is a “class” * the variable `a` is an object, or instance of the class * `append()` is a method

1.1.10 List Comprehensions

Python’s list comprehensions let you create lists in a way that is reminiscent of set notation

$$S = \{\sqrt{x} \mid 0 \leq x \leq 20,\ x \, mod \, 3 = 0\}$$

[32]:
```
import math
S = [math.sqrt(x) for x in range(20) if x % 3 == 0]
S
```

[32]:
```
[0.0,
 1.7320508075688772,
 2.449489742783178,
 3.0,
 3.4641016151377544,
 3.872983346207417,
 4.242640687119285]
```

[34]:
```
S = []
for i in range(2):
    for j in range(2):
        for k in range(2):
            S += [(i,j,k)]
S
```

[34]:
```
[(0, 0, 0),
 (0, 0, 1),
 (0, 1, 0),
 (0, 1, 1),
 (1, 0, 0),
 (1, 0, 1),
 (1, 1, 0),
 (1, 1, 1)]
```

[35]:
```
# you aren't restricted to a single for loop
S = [(i,j,k) for i in range(2) for j in range(2) for k in range(2)]
S
```

[35]:
```
[(0, 0, 0),
 (0, 0, 1),
 (0, 1, 0),
 (0, 1, 1),
 (1, 0, 0),
 (1, 0, 1),
 (1, 1, 0),
 (1, 1, 1)]
```

Syntax is generally

```
S = [<elt> <for statement> <conditional>]
```

1.1.11 Other Collections

We've seen the `list` class, which is ordered, indexed, and mutable. There are other Python collections that you may find useful: * `tuple` which is ordered, indexed, and immutable * `set` which is unordered, unindexed, mutable, and doesn't allow for duplicate elements * `dict` (dictionary), which is unordered, indexed, and mutable, with no duplicate keys.

[36]:
```
a_tuple = (1, 2, 4)
a_tuple[0] = 3
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_15332/1580541770.py in <module>
      1 a_tuple = (1, 2, 4)
----> 2 a_tuple[0] = 3

TypeError: 'tuple' object does not support item assignment
```

[14]:
```
a_set = {5, 3, 2, 5}
a_set
```

[14]:
```
[5, 3, 2, 5]
```

[38]:
```
a_dict = {}
a_dict[5] = 12
a_dict["key_2"] = 27
a_dict["key_3"] = [13, "value"]
a_dict
```

[38]:
```
{5: 12, 'key_2': 27, 'key_3': [13, 'value']}
```

1.1.12 Exercise 2

Lists 1. Create a list `['a', 'b', 'c']` 2. use the `insert()` method to put the element `'d'` at index 1 3. use the `remove()` method to delete the element `'b'` in the list

List comprehensions 1. What does the following list contain?

```
X = [i for i in range(100)]
```

2. Interpret the following set as a list comprehension: $S_1 = \{x \in X \mid x \mod 5 = 2\}$
3. Intepret the following set as a list comprehension: $S_2 = \{x \in S_1 \mid x \text{ is even}\}$