Python 101

Prashant Jamkhande

$ whoami

- Fellow member in HydPy
- Software engineer, Accenture
- Python and open source enthusiast

$ touch agenda

- Intro
- First basic things
- Data structures
- Conditional statements
- Looping

$ python3

Let’s begin to use Python interpreter

```
>>> # Let’s comment on what Python is ...a bit :)

>>> print(“Python is an interpreted language”)

>>> # Python as calculator

>>> import this

>>> # Declaring variables
```

Operator

- Mathematical: +, -, /, %, *
- Logical: and, or
- Relational: <, <=, >, >=, ==, !=

Few basic rules

- Use 4 spaces for indentation.
- Never mix tab and spaces.
- One blank line between functions.
- Two blank lines between classes.

Simple Task

Read a text file and print out non-comment lines

test.csv

```
#comment
Bank,Account,Amount
BAML,12345,$1000
"Citi Bank",54321,S$500
```

Pseudocode:

open file test.csv

for each line in file

if line does not start with ‘#’

print line

Python

```
with open('test.csv', 'r') as f:
    for l in f:
        if not l.startswith('#'):
            print l
```

Duck Type

Don't check whether something IS a duck. Check whether it QUACKS like a duck and WALKS like a duck

-- Alex Martelli

No need to specify data types

```
>>> a = 12

>>> type(a)

<class 'int'>

>>> a = 1.0

>>> type(a)

<class 'float'>

>>> a = “Hello world!”

>>>type(a)
```

Simple data types

- bool: Boolean, e.g. True, False
- int: Integer, e.g. 12, 23345
- float: Floating point number, e.g. 3.1415, 1.1e-5
- string: Character string, e.g. "This is a string"
- ...

Type conversion

float(string) -> float value

int(string) -> integer value

str(integer) -> string representation

str(float) -> string representation

```
>>> a = 8.126768

>>> str(a)

'8.126768'
```

Manipulate strings

- Concatenation using + operator
- Multiplaction using *

>>> ‘India’

>>> ‘India\’s best’

>>>”India’s best”

- Methods for string: split(), title(), upper(), swapcase(), isalnum() and more.

Read user input

input()

```
>>> help(input(“what help do you need?”))
```

How to print

```
>>> name = 'Sumith'

>>> country = 'India'

>>> print("%s is from %s" % (name, country))

Sumith is from India

>>> print("{0} is from {1}".format(name, country))

Sumith is from India

>>> print("{0} is from {0}".format(name))

Sumith is from Sumith
```

Multiple assignments

```
>>> a, b = 4, 5

>>> a

4

>>> b

5
```

Conditionals

```
if expression:
    do this
elif expression:
    do this
else:
    do this
```

Remember 4 space indentation?

```
if x == True:
    do this
if x == False:
    do this
```

Wrong way

```
if x:
    do this
if not x:
    do this
```

Right way

Looping

```
>>> while n < 11:
...     print(n)
...     n += 1
```

```
while condition:
    statement1
    statement2
```

For

```
for iterating_var in sequence:
    for iterating_var in sequence:
        statement
            statement
```

```
>>> for i in range(0, 11):  # Print 0-10
    >>> for i in range(0, 11):  # Print 0-10
...         print(i)
    ...         print(i)
```

```
>>> sum = 0
    >>> sum = 0
```

```
>>> n = 10
    >>> n = 10
```

```
>>> for i in range(1, n):
    >>> for i in range(1, n):
...         sum += 1
    ...         sum += 1
```

```
>>> print(sum)
    >>> print(sum)
```

Break and continue

```
>>> word = "Python3 Programming"
>>> for letter in word:
...     if letter == "3":
...         continue
...     print(letter)
```

Keywords

- and
- del
- from
- not
- while
- as
- elif
- global
- or
- with
- assert
- else
- if
- pass
- yield
- break
- except
- import
- print
- class
- exec
- in
- raise
- continue
- finally
- is
- return
- def
- for
- lambda
- try

Survival functions

- help()
- dir()

And…

Documentation : https://docs.python.org/3/

Structured data types

structured data types are composed of either simple or structured types

- List
- Tuples
- Set
- Dictionary
- strings

Slicing

[start, stop, step]

Applicable to any iterable collection of elements

Lists

- Lists are mutable
- List can be declared as :

L= list() or L=[]

- Lists are versatile. Since almost any data type can be added to them
- **List methods**
- List comprehensions:

```
squares = [x**2 for x in range(10)]
```

Tuples

Tuples are immutable.

Tuples are created in parentheses or using a comma

T=(1,2,3,4,4,5)

Concept of tuple packing and unpacking

Set

Sets are useful in fast membership check since no duplicate elements.

Creating set: set().

Uinon : a|b

Intersection : a&b

Diffrence : a-b

Symmetric difference : a^b #Letters in a or b but not in both

Set contains immutable objects however set itself is mutable

Dictionary

Creating dictionary: Dict={} or dict()

The values of a dictionary can be of any type but keys has to be of immutable data type

Memo: A previously computed solution can be used to avoid repetitive function call. Dictionaries help us to do this.

Dictionaries use hash values

Functions

```
def functionname(params):
    statement1
    statement2
>>> def sum(a, b):
...     return a + b
>>> res = sum(2, 3)
>>> res
```

References

1. http://pymbook.readthedocs.io/en/latest/
2. https://www.hackerrank.com/domains/python/py-introduction
3. http://projecteuler.net/
4.

Thank you!

Questions?

Reach out to me AT

prashantjamkhande@gmail.com