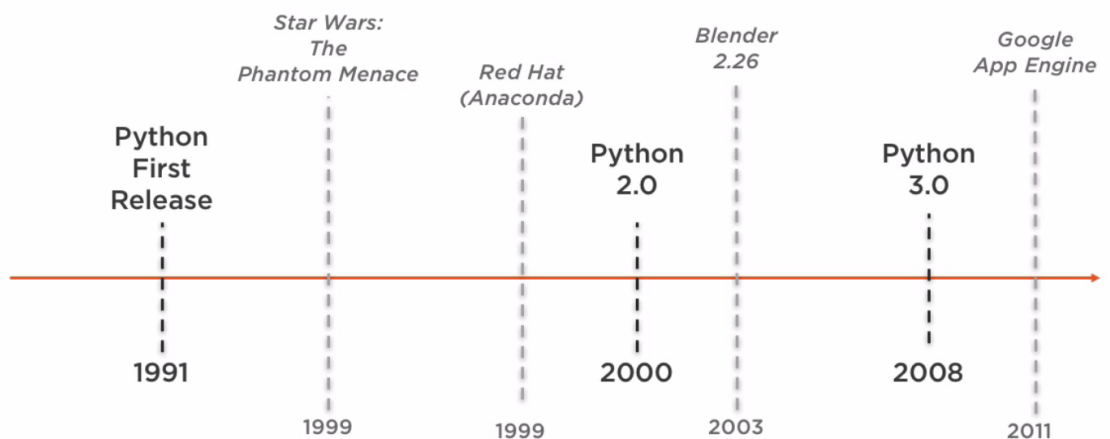


Python – Part 2

Python: The Big Picture

What Is Python?

Any type checks in python happen at run time instead of compile time. History of python



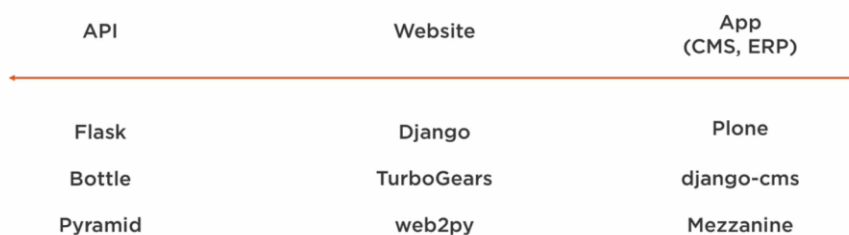
The python philosophy – PEP (python enhancement proposal) 20 – the zen of python – follow below principles



When and Where Is Python Being Used?

Python usage – Machine scripting and administration, web development, application scripting, data science.

Web Development



What Is Application Scripting?

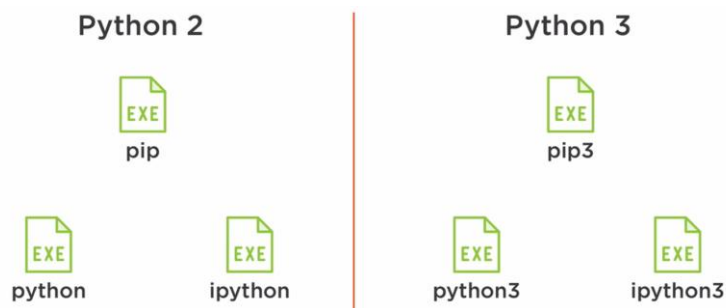


First Steps with Python

PIP means pip install packages. We pull the 3rd party libraries from pypi that is like the npm for python.

Continuing Your Python Journey

Python 2 vs. python 3. Python 2 is legacy code, we should use python 3 by default.



Ways to execute python code

- 1. Interpreter**
 - Executing python file using python exe
- 2. REPL**
 - Call out to Python code within interactive REPL
- 3. Natively**
 - "Compile and Run" (py2exe, pyinstaller, etc.)

Python: Variables, Data Types, and Conditionals

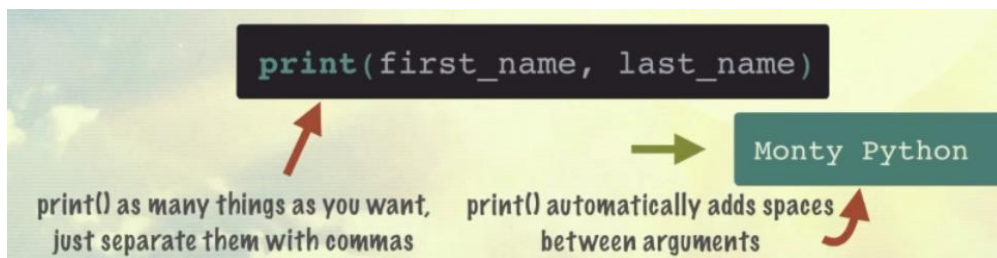
Birds & Coconuts

Naming of variable – use with underscore



Spam & Strings

We can give multiple arguments in print function:

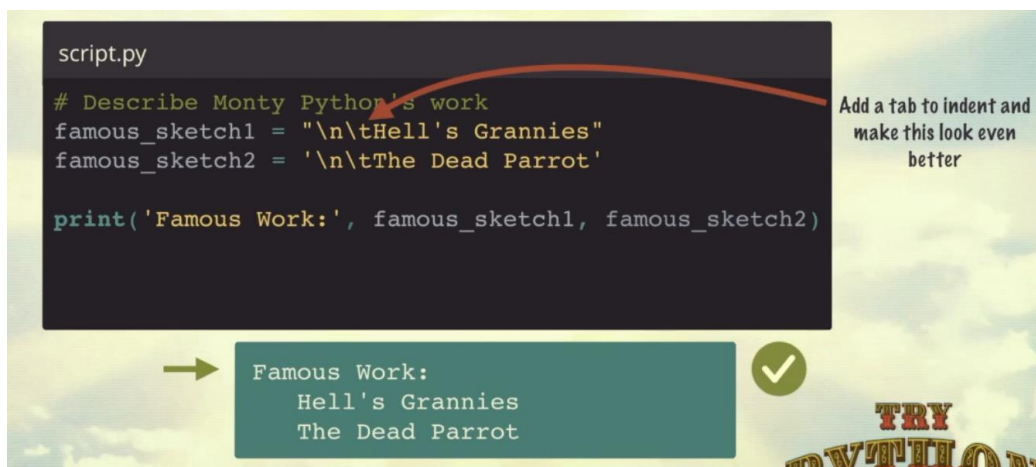


To write comment use # symbol.

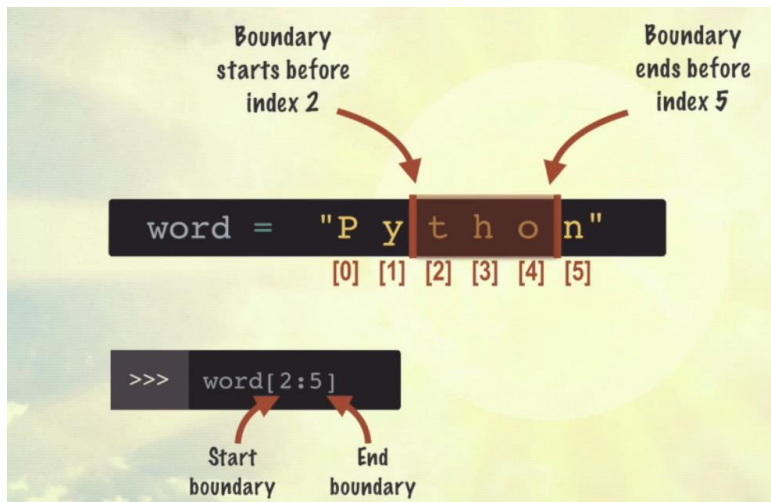
Convert a number into string



Using new line in print and for indent we can use \t



String inbuilt methods - `len()`, and use slice like below



Using integer division

// 2 division signs means integer division in Python.

half1 is $4//2 = 2$

half2 is $7//2 = 3$

```
script.py
# Calculate the halfway index
word1 = 'Good'
half1 = len(word1)//2  // Means integer division

word2 = 'Evening'
half2 = len(word2)//2  // Also rounds down to the nearest integer
```

'Good'

[0] [1] [2] [3]

'Evening'

[0] [1] [2] [3] [4] [5] [6]

Conditional Rules of Engagement

Conditional operators

There are 6 comparators in Python:

<	<=	==	>=	>	!=
less than	less than equal to	equal to	greater than equal to	greater than	not equal to

Using input function to take input from user

```
script.py
# Ask the user to input the number of knights
num_knights = int(input('Enter the number of knights'))
Change (or cast) the string to an int  User enters #, but it comes in as text

print('You entered:', num_knights)
if num_knights < 3 or day == 'Monday':
    print('Retreat!')
```

Python: Using Lists, Dictionaries, Loops, Files, and Modules

Lists and Dictionaries

Lists – append, remove, del

```
greetings = ['cheers', 'cheerio', 'watcha', 'hiya']
            index → [0]      [1]      [2]      [3]
```

```
>>> greetings[0] → 'cheers'
```

```
>>> greetings[3] → 'hiya'
```

```
>>> greetings[1:3] → ['cheerio', 'watcha']
```

You can also slice a list to get a sub-section of the list.

Dictionaries – maintaining two list can be error prone, so we can use dictionaries. Instead of using index we use key for lookup. For keys also we can use string, number. Methods available are – del. Use get method to get value even if doesn't exist, it won't throw an error.

key	value
'cheerio'	'goodbye'
'knackered'	'tired'

Great for our slang translator — we can look up the definition of a word with the word!

```
slang = {'cheerio': 'goodbye', 'knackered': 'tired', 'yonks': 'ages'}
```

Key Assign to Value

Each item is known as a "key-value pair"

```
print(slang['cheerio'])
```

→ 'goodbye'

To look up a value in a dictionary, we send in a key (instead of index in a list).

Comparing lists and dictionaries – for list the values should be same and in same order, but not required in dictionaries.

Instead of using lists of lists, prefer to use dictionary of lists for a better lookup –

A Dictionary of Lists

We can use a dictionary with keys for Breakfast, Lunch, and Dinner.

Dictionary where... each value is a list

```
menus = {'Breakfast': ['Spam n Eggs', 'Spam n Jam', 'Spam n Ham'],  
        'Lunch': ['SLT (Spam-Lettuce-Tomato)', 'PB&S (PB&Spam)'],  
        'Dinner': ['Spalad', 'Spamghetti', 'Spam noodle soup'] }  
  
print('Breakfast Menu:\t', menus['Breakfast'])  
print('Lunch Menu:\t', menus['Lunch'])  
print('Dinner Menu:\t', menus['Dinner'])
```

→ Breakfast Menu: ['Spam n Eggs', 'Spam n Jam', 'Spam n Ham']
Lunch Menu: ['SLT (Spam-Lettuce-Tomato)', 'PB&S (PB&Spam)']
Dinner Menu: ['Spalad', 'Spamghetti', 'Spam noodle soup']

Loops

Using range

```
for i in range(2005, 2016, 2):  
    print(i)
```

We can also call `range()` with more parameters: start, stop, and step.

→ 2005
2007
2009
2011
2013
2015

← Add 2

← Stop before 2016

Using dictionary's key and value in a loop, also using separator and format functions inside print

```
menu_prices = {'Knackered Spam': 0.50, 'Pip pip Spam': 1.50,  
              'Squidgy Spam': 2.50, 'Smashing Spam': 3.50}
```

```
for name, price in menu_prices.items():  
    print(name, ': $', format(price, '.2f'), sep='')
```

2 decimal places,
f for float format

We can format our price float to two decimal places using the built-in `format()` function.

→ Pip pip Spam: \$0.50
Squidgy Spam: \$1.50
Smashing Spam: \$2.50
Knackered Spam: \$3.50

✓ Looks great!

Functions

Use functions to avoid duplicate code, returning data from function is optional.

```

def average(numbers):
    total = 0
    for num in numbers:
        total = total + num

    avg = total/len(numbers)
    return avg

# Use our function on prices
prices = [2.50, 3, 4.50, 5]

result = average(prices)

print(result)

```

Program starts here: →

But inside the function, that data is accessed in a variable named numbers.

The prices array is sent as an input to the function.

For a better code, we should organize our main code into a main function

```

def average(numbers):
    total = 0
    for num in list:
        total = total + num

    avg = total/len(numbers)
    return avg

def main():
    prices = [29, 21, 55, 10]
    result = average(prices)
    print(result)

main()

```

Variables which are defined outside the function have global scope and their order also matters

```

def average(numbers):
    total = 0
    for num in list:
        total = total + num

    avg = total/len(numbers)
    return avg

def main():
    prices = [29, 21, 55, 10]
    result = average(prices)
    print(order_goal)
    print(result)

main()
order_goal = 25

```

At this point, order_goal hasn't been declared yet! So we'll get an error here...

We're calling main() before order_goal is declared...

TypeError: name 'order_goal' is not defined

Reading and Writing Files

Writing files

```
def write_sales_log(order):
    file = open('sales.txt', 'a') ← 'a' for append

    total = 0
    for item, price in order.items():
        file.write(item + ' ' + format(price, '.2f') + '\n')
        total += price

    file.write('total = ' + format(total, '.2f') + '\n')
    file.close()

def main():
    order = {'Cheeky Spam': 1.0, 'Yonks Spam': 4.0}
    write_sales_log(order)
    order = {'Cheerio Spam': 1.0, 'Smashing Spam': 3.0}
    write_sales_log(order)
```

Reading file – read(), readline(), we can assign it into list. Also use string method strip to remove leading and trailing whitespace

```
def read_dollar_menu():
    dollar_spam = open('dollar_menu.txt', 'r')

    dollar_menu = []
    for line in dollar_spam:
        dollar_menu.append(line) ← This will combine all
                                   items into a list.

    print(dollar_menu)
    dollar_spam.close()
```

Exceptions

```
price = input("Enter the price: ")

try:
    price = float(price)
    print('Price =', price)
except ValueError: ← We can also look for a
                    specific type of error.
    print('Not a number!')
```

→ Enter the price: 25
Price = 25.0

The user enters a number so there's no error.

→ Enter the price: oops
Not a number!

Program prints the exception, telling the user what the problem was, then continues.

Modules

Modules are code libraries that contain functions we can call from our code. They make our lives easier by implementing the hard stuff for us. We can use pip to install modules.

Instead of rewriting complicated code,
it's already been done for us!

```
import random

ticket = random.randint(1, 1000)
print(ticket)
```

Reuse code instead of reinventing the wheel!

```
import math

answer = math.sqrt(3)
print(answer)
```

The http module

```
import requests
my_request = requests.get('http://go.codeschool.com/spamvanmenu')
menu_list = my_request.json()
print(menu_list)
```

Import the module so we can use it for requests
 Call requests.get() with our URL
 Get the response in JSON format

```
[{"price": "3.00", "name": "Omelet", "desc": "Yummy"},
 {"price": "5.75", "name": "Burrito", "desc": "Breakfast Burrito"},
 {"price": "4.50", "name": "Waffles", "desc": "Belgian waffles with syrup"}]
```

Creating modules – a script containing definitions is a module and can be imported by a script or another module.

spam_van.py
 import orders
 import menu
 import sales

orders.py
 Module

menu.py
 Module

sales.py
 Module

Script

same
directory

As long as all of these files are in the same directory, we can import them into spam_van.py by writing `import module_name`.

We need to add the module name `orders.` before our function calls

```
orders.py
def print_menu(menu):
...
def get_order(menu):
...
def total_bill(orders, menu):
...

spam_van.py
import orders

def main():
    menu = {'Cheerio Spam': 0.50,...}
    orders.print_menu(menu)
    orders = orders.get_order(menu)
    total = orders.bill_total(orders, menu)
    print('You ordered:', order, 'Total:', total)

main()
```