Introduction to python

Charles Cao

ccao@ecvictor.com

Thanks to www.programiz.com

Instructors

- Charles Cao
  - Team leader of Paysafe
  - Founder of ECVictor
  - 10+ software development experience
- Dr. Tony Deng
  - Data scientist
- Richard Yan
  - Senior python developer

Python Salary

Salary | Hourly Rate | Bonus | Benefits | More

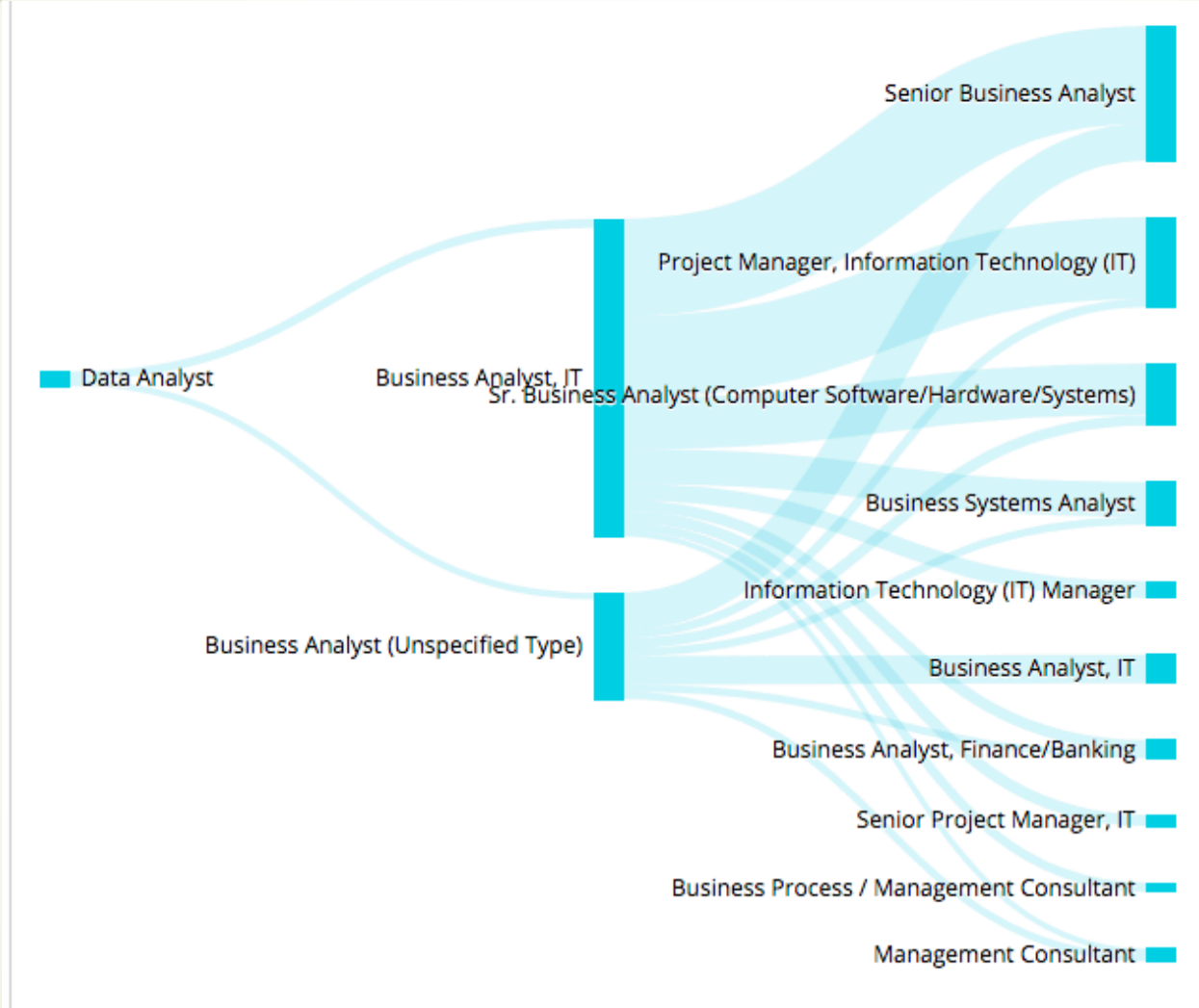
Python Median Salary by Job

Check salary info for your own job »

More Charts

Job	Average	Min	Max
Software Developer 244 profiles	C$62,090	C$45K	C$89K
Software Engineer 189 profiles	C$72,009	C$52K	C$94K
Sr. Software Engineer / Developer / Programmer 73 profiles	C$88,559	C$60K	C$111K
Senior Software Engineer 56 profiles	C$96,866	C$78K	C$123K
Data Scientist 49 profiles	C$75,031	C$50K	C$103K
Software Engineer / Developer / Programmer 45 profiles	C$65,368	C$44K	C$88K
Data Analyst 39 profiles	C$59,223	C$41K	C$83K

Common Career Paths for Data Analyst



Course Outline

- *Hello, World!*
  Variables and Types
  Lists
  Basic Operators
  String Formatting
  Basic String Operations
  Conditions
  Loops
  Functions
  Classes and Objects
  Dictionaries
  Modules and Packages

Projects

- *Automation Testing*
  Selenium Automation testing with Python

- *Stock Market Prediction in Python*
  The aim of the project is to predict whether future daily returns of a S&P 500 are going to be positive or negative.

- *Bitcoin/Digital currency Automated Trading*
  Professional algorithmic trading solution to support automated Bitcoin/Digital currency trading
  Reference:
  http://francescopochetti.com/stock-market-prediction-part-introduction/

An Informal Introduction to Python

History of Python

- **Why Python was created?**
- In late 1980s, Guido Van Rossum was working on the Amoeba distributed operating system group. He wanted to use an interpreted language like ABC (ABC has simple easy-to-understand syntax) that could access the Amoeba system calls. So, he decided to create a language that was extensible. This led to a design of new language which was later named Python.

Release Dates of Different Versions

Python 1.0 (first standard release)	January 1994
Python 1.6 (Last minor version)	September 5, 2000
Python 2.0 (Introduced list comprehensions)	October 16, 2000
Python 2.7 (Last minor version)	July 3, 2010
Python 3.0 (Emphasis on removing duplicative constructs and module)	December 3, 2008
Python 3.6.2 (Last updated version)	December 12, 2016

Run Python on Your Operating System

- https://www.python.org/downloads/release/python-362/
- When the download is complete, open the package and follow the instructions. You will see "The installation was successful" message when Python is successfully installed.

Hello World Example

- print("Hello world!")

Python Keywords

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

We cannot use a keyword as variable name, function name or any other identifier. They are used to define the syntax and structure of the Python language. In Python, keywords are case sensitive.

Python Statement

```
a = 1 + 2 + 3 + \
    4 + 5 + 6 + \
    7 + 8 + 9
```

```
a = 1; b = 2; c = 3
```

Python Comments

- #This is a comment
- #print out Hello
- print('Hello')

- #This is a long comment
- #and it extends
- #to multiple lines

Variables and Types

- **Numbers**
  - Python supports two types of numbers - integers and floating point numbers.
  - **E.g A=7**
- **Strings**
  - Strings are defined either with a single quote or a double quotes.

Mixing operators between numbers and strings is not supported

- one = 1
- two = 2
- hello = "hello"
- print(one + two + hello)

Naming Styles

- b (single lowercase letter)
- B (single uppercase letter)
- lowercase
- lower_case_with_underscores
- UPPERCASE
- UPPER_CASE_WITH_UNDERSCORES
- CapitalizedWords (or CapWords, or CamelCase -- so named because of the bumpy look of its letters. This is also sometimes known as StudlyCaps.
- Note: When using abbreviations in CapWords, capitalize all the letters of the abbreviation. Thus HTTPServerError is better than HttpServerError.
- mixedCase (differs from CapitalizedWords by initial lowercase character!)
- Capitalized_Words_With_Underscores (ugly!)

Tip

- local_var_name
- GLOBAL_VAR_NAME
- module_name
- package_name
- ClassName
- method_name
- ExceptionName
- function_name
- instance_var_name
- function_parameter_name

Lists

- Lists are very similar to arrays
  - mylist = []
  - mylist.append(1)

List count start from 0

- my_list=[1,2,3,4]
- print(1)

Basic Operators

- Arithmetic Operators
- Comparison (Relational) Operators
- Assignment Operators
- Logical Operators
- Bitwise Operators
- Membership Operators
- Identity Operators

Arithmetic Operators

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	a + b = 30
- Subtraction	Subtracts right hand operand from left hand operand.	a – b = -10
* Multiplication	Multiplies values on either side of the operator	a * b = 200
/ Division	Divides left hand operand by right hand operand	b / a = 2
% Modulus	Divides left hand operand by right hand operand and returns remainder	b % a = 0
** Exponent	Performs exponential (power) calculation on operators	a**b =10 to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	9//2 = 4 and 9.0//2.0 = 4.0, -11//3 = -4, -11.0//3 = -4.0

Comparison Operators

Operator	Description	Example
==	If the values of two operands are equal, then the condition becomes true.	(a == b) is not true.
!=	If values of two operands are not equal, then condition becomes true.	
>	If the value of left operand is greater than the value of right operand, then condition becomes true.	(a > b) is not true.
<	If the value of left operand is less than the value of right operand, then condition becomes true.	(a < b) is true.
>=	If the value of left operand is greater than or equal to the value of right operand, then condition becomes true.	(a >= b) is not true.
<=	If the value of left operand is less than or equal to the value of right operand, then condition becomes true.	(a <= b) is true.

Assignment Operators

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a
/= Divide AND	It divides left operand with the right operand and assign the result to left operand	c /= a is equivalent to c = c / ac /= a is equivalent to c = c / a
%= Modulus AND	It takes modulus using two operands and assign the result to left operand	c %= a is equivalent to c = c % a
	Performs exponential (power)	

Logical Operators

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

Membership Operators

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

Identity Operators

Operator	Description	Example
is	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	x is y, here **is** results in 1 if id(x) equals id(y).
is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here **is not** results in 1 if id(x) is not equal to id(y).

Bitwise Operators

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
| Binary OR	It copies a bit if it exists in either operand.	(a | b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	a << 2 = 240 (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	a >> 2 = 15 (means 0000 1111)

Operators Precedence

Operator	Description	
**	Exponentiation (raise to the power)	
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)	
* / % //	Multiply, divide, modulo and floor division	
+ -	Addition and subtraction	
>> <<	Right and left bitwise shift	
&	Bitwise 'AND'	
^ |	Bitwise exclusive `OR' and regular `OR'	
<= < > >=	Comparison operators	
<> == !=	Equality operators	
= %= /= //= -= += *=		