

Python Intro

Christoffer Berglund

September 6, 2018

Overview

- Repetition powershell (and bash)
- Game
- Python
- Arrays
- Running other applications
- Help and dir
- Rich libraries

Repetition powershell

Variables

Type inference

Bash:

```
echo "Enter a number"
read a
b=25
let "c=$a+$b"
```

Powershell:

```
$a = Read-Host 'Enter a Number'
$b = 25
$c = ([int]$a+$b)
```

Variables from programs

```
$a = $(ls)
```

Pipes and streams

Pipes, Streams and Redirection - Exactly like Bash

```
ls | more
ls > file
ls 1> file
ls 2> error
```

Grep, Sort, Head and Tail

Grep

bash:

```
ls -alh | grep ".txt"
```

Powershell:

```
ls | where {$_ -match ".tex"}
```

or

```
ls | where {$_.Name -like "*.tex"}
```

- like demands exact match
- match is text contains

Sort

```
ls | sort -p length
```

Head and tail

head:

```
ls | sort -p length | select -first 5
cat textfile | select -first 5
```

tail:

```
ls | sort -p length | select -last 5
cat textfile | select -last 5
```

head and tail

```
ls | select -last 2 -first 2
```

Functions

Declaring a function

Bash:

```
function addn{
    echo $a +$b
}
```

Powershell:

```
function addn($a, $b){
    return $a + $b
}
$c = addn 4 5
```

Bash-like functions

```
function addn{
    echo $a + $b
}
```

- Wrong

5 Minute Assignment

- Write a rot13 function encrypt a single character work as following:
 - Change the character to char, then to int. (translate from letter to ascii value)
`[int] [char] $a`
 - Decrement 13
 - Change the letter to char and return it.

Solution

```
function encrypt($a){  
    $number = [int][char]$a  
    $number=$number+13;  
    return [char]$number  
}
```

Conditions

if

```
if(STATEMENT){  
    echo "If-statement is true"  
}  
elseif(STATEMENT){  
    echo "else-if statement is true"  
}  
else{  
    echo "None of the above are true"  
}
```

if

```
$alder = 18  
if($alder -lt 18){  
    echo "Ingen øl på deg"  
}  
elseif($alder -eq 18){  
    echo "Akkurat gammel nok"  
}  
else{  
    echo "Gammel nok"  
}
```

Comparisons

Operator	Meaning
-lt	Less than
-gt	Greater than
-le	Less than or equal to
-ge	Greater than or equal to
-eq	Equal to
-ne	Not equal to

Boolean

Operator	Meaning
-not	Not
!	Not
-and	And
-or	Or

5 Minute Assignment

- Write a script that asks the user for a new password.
- If the length of the password is less than 8, say that it too short.
- If the length of the password is more than 20, say that it is too long.
- Otherwise, say just right.
- Hint to check the length of the string variable \$a, use \$a.length

Solution

```
$pwd=Read-Host 'Enter a new password'
if($pwd.length -lt 8){
    echo "Too short"
}
elseif($pwd.length -gt 20){
    echo "Too long"
}
else{
    echo "Just right"
}
```

Arrays

Arrays

```
$machines = @("www.uia.no","grimstad.uia.no")
echo $machines
echo $machines[0]
```

For loop over arrays

```
$machines = @("www.uia.no","grimstad.uia.no")
echo $machines
#echo $machines[0]

foreach($i in $machines){
    echo $i
}
```

Associative Arrays (dictionaries)

```
$machines = @{
    "www.uia.no" = "158.36.166.145";
    "grimstad.uia.no" = "158.36.52.10"
}
echo $machines["www.uia.no"]
echo $machines.keys
echo $machines.values
```

Command line arguments

- Command line arguments are arrays

```
echo $args
echo $args[0]
```

Structs and classes

Structs, Classes and Objects

- You can Classes, Objects and Inheritance as you do with any .NET language.
- Read more at e.g. <https://docs.microsoft.com/en-us/powershell/scripting/powershell-scripting?view=powershell-6>

Alias and Environment

Alias

- Mapping a different cmdlet

```
set-alias sz "c:\Programfiler\7-Zip\7z.exe"
```

env

- Many environment variables are available on Windows

```
dir env:  
echo $env:ProgramFiles  
echo $env:HOMEPAH
```

5 Minute Assignment

- Following is some code automatically copies and zips down every powershell file in a directory.
- Take some minutes to read and understand the code.
- Update the solution to use alias and env

```
cd c:\Users\mortengo\powershell  
$a = $(ls)  
mkdir "homework"  
#$a = Get-ChildItem "c:\Users\mortengo\powershell"  
  
foreach($f in $a){  
    if($f -match ".ps1"){  
        echo $f  
        cp $f ./homework  
    }  
}  
  
& 'c:\Program files\7-Zip\7z.exe' a -tzip homework.zip homework
```

Solution

```
set alias zip "$env:ProgramFiles\7-Zip\7z.exe"  
$homeworkdir="$env:HOMEPAH" +"/powershell"
```

```

cd $homeworkdir
$a = $(ls)
mkdir "homework"
#$a = Get-ChildItem $homeworkdir

foreach($f in $a){
    if($f -match ".ps1"){
        echo $f
        cp $f ./homework
    }
}
zip a -tzip homework.zip homework

```

Game

Bash or Powershell?

```

a="Harry Potter Rules"
a=${a/Harry/Star}
a=${a/Potter/Wars}
echo $a

```

awnser: Bash

Bash or Powershell?

```

function best($a,$b){
    if($a -gt $b){
        return $a
    }
    else{
        return $b
    }
}

```

awnser: PowerShell

Bash or Powershell?

```

a=12
b=16

```



```
if [ $a -gt $b ]; then
    echo $a
else
    echo $b
fi
```

awnser: Bash

Bash or Powershell?

```
a=12
b=16
if($a -gt $b){
    echo $a
}
else{
    echo $b
}
```

awnser: Bash

Bash or Powershell?

```
a=12
b=16
if ( $a -gt $b ); then
    echo $a
else
    echo $b
fi
```

awnser: Trick question, none of them

Python



What is Python

- Scripting language
 - As with Bash and Powershell: You do not compile the entire file but run line for line.
 - Can program in an interpreter

Python versus Java



Designer: Guido van Russom

- Idea behind Python
 - Easy and intuitive.
 - * Just see how easy it is compared to bash
 - * Understandable if you just speak English
 - Short development time

Easy to understand

```
if(age>32):
    print("You are old")
else:
    print("You are young")
```

Short: Password generator in Python

```
''.join([chr(random.randint(64,128)) for i in range(10)])
```

Equivalent in Java

```
public class PassordGenerator{
    public String getRandomPassword() {
        StringBuffer password = new StringBuffer(20);
        int next = RandomUtils.nextInt(13) + 8;
        password.append(RandomStringUtils.randomAlphanumeric(next));
        return password.toString();
    }

    public static void main(String[] args){
        PassordGenerator pg = ne PassordGenerator();
        pg.getRandomPassword();
    }
}
```

Download Python

- Very different from version 2 and 3.
 - Short story: Python 2.x is legacy, Python 3.x is the present and future of the language
 - <https://wiki.python.org/moin/Python2orPython3>

- We use mainly 3.
 - * Could be some libraries are only available in 2.
- <http://www.python.org/getit/>
- 3.6.2
- 2.7.13

Reserved words

Interpreter vs Compiler

- Interpreter interprets instructions on the fly
- Python interpreter is interactive
 - Write a line of code
 - Python process immediately
 - Python wait for new command
- Interactive conversation
- Compiler must know the whole program
- Translate to static machine language for later execution
- Creates *.exe or *.dll in Windows

Python implementation

- Python implemented in C
- C does not support OO fully
- Python support OO functional features

Python pip package managers

- pip - package manager to install and manage software packages written in python.

```
$ pip install <package> # installs packages from the web
$ pip install -r requirements.txt # installs packages listed in given txt file
$ pip install <local package>.whl # installs local wheel packages
$ pip install --upgrade <package> # upgrades package
$ pip freeze # list installed packages
```

Python Anaconda (conda) package manager

- Anaconda - package manager, environment manager, Python distribution
 - Contains more than 700 packages
 - Miniconda - Miniconda
- install more packages under Anaconda?

```
$ conda install
```

Python Homebrew package manager

- The missing package manager for macOS

Pipenv and Virtual environment

- Check Python version

```
$ python --version
```

- pipenv is a "dependency manager for Python projects... high level tool that simplifies dependency management for common usecases"
- virtualenv - tool to create isolated Python environments. It "creates a folder which contains all the necessary executables to use the packages that Python projects would need."
- virtualenvwrapper - extensions to virtualenv and makes working with virtualenv more pleasant. Places all your virtual env. in one place.

Virtualenvwrapper

- install virtualenvwrapper using apt-get:
- Test your installation:

```
$ virtualenvwrapper
```

- create a virtual environment

```
$ mkvirtualenv name_of_virtual_env
```

- creates a folder and copies all needed files, including python.exe and pip for the given version.

Virtualenvwrapper usage

- activate the virtual environment (linux)

```
$ workon name_of_virtual_env
```

- freeze your environment (creating a text file with a list of packages in current environment with their respective versions.)

```
$ pip freeze > requirements.txt
```

- deactivate the virtual environment

```
$ deactivate
```

- remove a virtual env

```
$ rmvirtualenv name_of_virtual_env
```

- set python version

```
$ mkvirtualenv --python=`which python2` name_of_virtual_env
```

Variables

- Type inference as in Bash

Bash:

```
age=32
echo $age
```

Python:

```
age = 32
text = "Hei"
print(age)
```

Any math

Bash:

```
a=12
b=13
let "c=$a+$b"
```

Python:

```
a=12
b=13
c=a+b
```

Reading from users

Bash:

```
echo "Enter a number"
read a
```

Python

```
a = input("Enter a number")
```

Forcing ints

```
a = int(input("Enter a number"))
a=a+2
```

or

```
a = input("Enter a number")
a=int(a)+2
```


5 Minute Assignment

- Make a simple calculator that asks the user for two numbers.
- The program should print the sum.

Solution

```
a = int(input("Number 1:"))
b = int(input("Number 2:"))
c = a+b
print(c)
```

Supports functionality

- But not a functional language.
- (Almost) everything are functions.

Python 2:

```
alder = 32
print alder
```

Python 3:

```
alder = 32
print(alder)
```

Indenting and ifs

- No curly braces

```
if(age>32):
    print("You are old")
else:
    print("You are young")
```

Indenting and ifs (2)

- No curly braces

```

if(age>32):
    print("You are old")
elif (age<12):
    print("You are too young")
else:
    print("You are just right")

```

5 Minute Assignment

- Write a script that asks the user for a new password.
- If the length of the password is less than 8, say that it too short.
- If the length of the password is more than 20, say that it is too long.
- Otherwise, say just right.
- Hint: to read from the user `raw\input`
- Hint: to check the length of the string variable use `len(password)`

Solution

```

password = input('Enter password:')
if(len(password)>20):
    print("Too long")
elif (len(password)<8):
    print("Too short")
else:
    print("Just right")

```

Arrays

List / Arrays

Bash (array):

```

os=('linux' 'mac')
for i in "${os[@]}"
do
    echo $i
done

```

Python (list):

```
os = ['linux', 'mac']  
for i in os:  
    print(i)
```

Making lists

Splitting lists:

```
ip = '192.168.1.1'  
l = ip.split('.')
```

Getting one item in a list:

```
l[0]
```

Getting more than one item (slicing)

```
l[0:3]
```

5 Minute Assignment

- You have the text "This is a long sentence"
- Split the sentence and print out every word in a for loop
- Start with the following code:

```
s = "This is a long sentence"
```

Solution

```
s = "This is a long sentence"  
l = s.split(' ')  
for i in l:  
    print(i)
```

For loops

```
tekst = "abcdef"  
for bokstav in tekst:  
    print(tekst)
```

or

```
for i in range(0,10):  
    print(i)
```

Double loops

```
for i in 'ab':
    for j in 'ab':
        print(i,j)
```

5 Minute Assignment

- Create a small application that generates every 4-digit pin code available.

Solution

```
for i in range(0,10):
    for j in range(0,10):
        for k in range(0,10):
            for l in range(0,10):
                print(i,j,k,l)
```

Dictionaries (Associative Arrays)

Bash:

```
declare -A user
user=( [chrisb]="Christoffer Berglund" [sigurda]="Sigurd Assev")
for i in "${!user[@]}"
do
    echo "key : $i"
    echo "value: ${user[$i]}"
done
```

Python:

```
passwordlist = {'chrisb':'ind2978e', 'sigurda':'hemmelig123'}
for user,password in passwordlist.items():
    print(user)
    print(password)
```

Using dictionaries

```
passwordlist = {'chrisb':'ind2978ey', 'sigurda':'hemmelig123'}
username = input('Skriv ditt brukernavn')
print("Ditt passord er",passwordlist[username])
```

5 Minute Assignment

- Create a dictionary with two/three Norwegian and English words.
- Let the user enter a Norwegian word and translate it to English.

Solution

```
ordbok = {'hei': 'hi', 'bil': 'car', 'rom': 'room'}  
norsk = input('Skriv norsk ord')  
print ("Det engelske ordet er:", ordbok[norsk])
```

Running other applications

Running other application

Bash:

```
ls
```

Python:

```
import os  
os.system("dir")
```

Grabbing stdout from application

bash:

```
a=$(ls)  
echo $a
```

python:

```
p = subprocess.Popen('ls', stdout=subprocess.PIPE)  
d = p.stdout.readline()  
while(d!=b''):  
    print(d)  
    d = p.stdout.readline()
```

- This will not work in Windows!

Help and dir

Need help

Bash:

```
man ls
```

```
import random
help(random)
dir(random)
```

Rich libraries

Easy use and rich libraries

```
import random
tall = random.randint(0,10)
```

- Many "hacking" libraries: mechanize, pexpect, urllib,
- Bash does not have libraries.

Using random

```
tall = random.randint(64,128)
bokstav = chr(tall)
```

5 Minute Assignment

- Write code that generates a password of length 10

Solution

```
for i in range(10):
    r = random.randint(64,128)
    print(chr(r))
```

Read from a web site

```
from urllib.request import urlopen
html = urlopen('http://vg.no')
print(html.read())
```

Read from a website and change everything to uppercase

- Curriculum later

```
from urllib.request import urlopen
html = urlopen('http://vg.no')
data = str(html.read())
data = data.upper()
open('data.html', 'w').write(data)
```

5 Minute Assignment

- Make a program that downloads the uia homepage

Read from a web site

```
from urllib.request import urlopen
html = urlopen('http://uia.no')
print(html.read())
```

Hand-Raising Assignment

- Python-kode. Oppdag feil:

```
$a = int(input('skriv et tall'));
if(a>12)
    print "Du har skrevet mer enn 12"
fi
pinkoder = ['morten': '1309', 'sigurd': '1234']
print("Pinkoden til Morten er", pinkoder['morten'])
```

err: \$a, semicolon, parantes print, ending fi, mix of dict and list, last print it messed up