Pink Programming

Python Camp

Code Handouts

PINK PROGRAMMING

How to read

The handouts are meant to match the slides. They are also meant as a practical reminder. **You would need to run the code to see the output.**

run in **commandline**

run in **interpreter**

concepts - not pure code

write in a python file then execute file

output

input

This handout belongs to:

_____

Install and “Hello, World!”

Checking If you have python and pip installed

```
python3 -V
pip3 -V
```

Example output

```
$ python3 -V
Python 3.6.1
$ pip3 -V
pip …
```

If it does not work, download and install python here: www.python.org/downloads/

Python Interpreter - an interactive commandline tool to try out python code

```
python3
```

You will now be in the interpreter.

```
print("Hello, world!")
exit() # or quit()
```

Command Line - Mac/Unix

Basic commands

```
cd path  #Changing a folder
ls -la #Listing folder contents
pwd #Listing your current location
mkdir folder_name #Making a folder
touch filename.txt #Creating a file
cat filename.py #print file content
```

Special folders

```
/  the root of your system
~  your user's home
.  current directory
..  parent directory
```

Tips:
Use **Tab** for autocomplete
Windows users can use this with:

- https://cygwin.com/

Command Line - Windows

Note!!! In windows, the **interpreter** is opened with:

```
python
```

Basic commands

```
cd path  #Changing a folder
dir #Listing folder contents
cd #Listing your current location
mkdir folder_name #Making a folder
echo "" > filename.txt #Create file
type filename.py #print file content
```

Special folders

```
c:\  the root of your system
.  current directory
..  parent directory
```

Tip:
Use **Tab** for autocomplete
Use **** instead of **/**

Data types

Python data types

`int` *(integer)* - numbers: `-3`, `0`, `23`
`float` - decimal numbers: `-34.5`, `-2.0`, `3.98`
`string` - words: `"game"`, `'Game'`, `"SAME"`
`bool` *(boolean)*: `True`, `False`
`None`

Strings

```
print("Game Over")
print('Game Over')
print("Game", "Over")
print("Here", end = " ")
print("it is...")
```

Escaping sequences with Strings

New line - **`\n`**
Tab - **`\t`**
Backslash - **`\\`**
Inserting a Quote - **`\'`** **`\"`**

```
print("a\nb")
print("c\td")
print("e\\f")
print("g\"h")
print("\a")
```

Concatenating Strings

```
print("snow" + "ball")
print("Pie" *10)
```

Variables

Simple create

```
name = "Nadia"
print(name)
print("Hi,", name)
```

Naming variables

Only **numbers, letters** or _
Can't start with a number
Descriptive names score not sc
Be consistent snow_ball not snowBall
Not too long (under 15 char)

Mathematical Operators

```
2 + 3  # Addition / Plus
4 - 6  # Subtraktion / Minus
2 * 5  # Multiplication
5 / 2  # Division
5 // 2 # Division
5 % 2  # Modulus
```

Augmented Assignment Operators

```
x = 10 ; print(x)
x *= 5 ; print(x) # x=x*5
x /= 5 ; print(x) # x=x/5
x %= 5 ; print(x) # x=x%5
x += 5 ; print(x) # x=x+5
x -= 5 ; print(x) # x=x-5
```

User input

```
name = input("Hi! What's your name?")
print("Your name is: " + name)
```

String methods

```
name = "NadIa"
name.upper()
name.lower()
name.title()
name.replace("d","tal")
```

Converting types

```
int("42")
float("3.42")
str(3)
age = int(input("How old are you?"))
print("Your age is: ", age)
```

Comparison operators

```
5 == 5  # equal-to
8 != 5  # not equal to
3 > 10  # greater than
5 >= 10 # greater than or = to
5 <  8  # less than
5 <= 5  # less than or = to
```

If statement

```
password = "secret"
if password == "secret":
    # note the indentation
    print("Access granted!")
```

if - elif - else

```
weekday = 1
if weekday == 1:
    print("It's Monday!")
elif weekday == 2:
    print("It's Tuesday!")
else:
    print("It's Weekend!!!")
```

While loop

Using condition

```
response = ""
while response != "Because":
    response = input("Why?")
```

Using counters

```
counter = 0
while counter <= 10:
    print(counter)
    counter += 1
```

Break and continue

```
count = 0
while True:
    count += 1
    if count > 10:   #end if count > 10
        break        #to exit a loop
    if count == 5:   #skip 5
        continue     #to jump back
    print(count)
```

Multiple conditions

```
cookies = True
hungry = True
if cookies and hungry:
    print("Fika time!")
```

```
cookies = True
hungry = False
if cookies and not hungry:
    print("No fika yet!")
```

and, or and not

a	b	a **and** b	a **or** b	**not** a	**not** b
True	True	True	True	False	False
True	False	False	True	False	True
False	True	False	True	True	False
False	False	False	False	True	True

Importing a module

```python
import random
#produces a random integer between 10-40
print(random.randint(10, 40))
#random integer between 0 and 10
print(random.randrange(10))
```

For loops

```python
for i in range(0, 3):
    print("i1 = ", i)
for i in range(0, 50, 5):
    print("i2 = ", i)
for i in range(50, 0, -5):
    print("i3 = ", i)
```

Index

0	1	2	3	4
i	n	d	e	x

String as a sequence

```python
word = "programming"
print(word[0])
print(word[5])
```

```python
for i in "index":
   print(i)
```

Lists

Sequence - like string, but can contain anything you want

index	0	1	2	3	4	5
	"cat"	"dog"	"fish"	"cow"	"pig"	"fox"

```
animals = ["cat", "dog", "fish", "cow", "pig", "fox"]
```

Accessing elements

```
animals = ["cat", "dog", "fish", "cow",
"pig", "fox"]
print(animals[2])
print(animals[1:4])
print(animals[:3])
print(animals[4:])
print(len(animals))
```

Concatenating lists

```
farm_animals = ["sheep", "cow", "pig"]
wild_animals = ["fox", "wolf", "eagle"]
all_animals = farm_animals + wild_animals
print(all_animals)
```

Negative index

0	1	2	3	4
i	n	d	e	x
-5	-4	-3	-2	-1

```python
word = "programming"
print(word[-1])
print(word[-3:])
print(word[:-2])
print(word[-4:-2])
```

Lists

Deleting

```python
all_animals = ["sheep", "cow", "pig",
"fox", "wolf", "eagle"]
del all_animals[5]
print(all_animals)
del all_animals[-1]
print(all_animals)
```

Appending

```python
all_animals = ["sheep", "cow", "pig",
"fox", "wolf"]
all_animals.append("horses")
print(all_animals)
```

Change a value

```python
all_animals = ["sheep", "cow", "pig",
"fox", "wolf", "horses"]
all_animals[-1] = "horse"
print(all_animals)
```

Loop

```python
all_animals = ["sheep", "cow", "pig",
"fox", "wolf", "horse"]
for animal in all_animals:
    print(animal)
```

Lists in lists

```
farm_animals = ["sheep", "cow", "pig",
"horse"]
wild_animals = ["fox", "wolf", "eagle"]
all_animals = [farm_animals, wild_animals]
print(all_animals)
```

Dictionaries

List of key-value pairs

```
animal_info = {"sheep": "stubborn", "cow": "gives milk"}
```

key value key value

Try it

```
animal_info = {"sheep": "stubborn",
               "cow": "gives milk"}
print(animal_info["sheep"])
```

Adding an item

```
animal_info = {"sheep": "stubborn",
               "cow": "gives milk"}
animal_info["wolf"] = "wants to eat cow"
print(animal_info)
```

Deleting an item

```
animal_info = {"sheep": "stubborn",
               "cow": "gives milk",
               "wolf": "wants to eat cow"}
del animal_info["wolf"]
print(animal_info)
```

Dictionary loop

```
animal_info = {"sheep": "stubborn",
               "cow": "gives milk"}
for key in animal_info:
    print(key)
    print(animal_info[key])
```

```
animal_info = {"sheep": "stubborn",
               "cow": "gives milk"}
for key, value in animal_info.items():
     print(key + " has info: " + value)
```

Functions

You've actually already used them!

- print(<string>)
- len(<list>)
- range(<from>, <to>, <step>)
- ...

Writing a function

name parameter

```
def circle_circumference(radius):
""" Return circle circumference """
```

docstring

```
    circumference = 2 * 3.14 * radius
    return circumference
```

return value

Try it

```
def circle_circumference(radius):
    """ Return circle circumference """
    circumference = 2 * 3.14 * radius
    return circumference
my_circumference = circle_circumference(5)
print(my_circumference)
```

Note!!! Functions - can be reused. That is why we return the value, to be used by the code calling it.

Functions import

from <filename> **import** circle_circumference

Try it

```
from math import pi
def circle_circumference(radius):
    """ Return circle circumference """
    circumference = 2 * pi * radius
    return circumference
my_circumference = circle_circumference(5)
print(my_circumference)
```

Parameters

```
def birthday_wishes1(name, age):
    print("Happy birthday, " + name +
    "! You're " + str(age) + ".\n")
birthday_wishes1("Ana", 10)
```

```
# Positional parameters
def birthday_wishes2(name = "Ana",
                     age = 12):
    print("Happy birthday, " + name +
        "! You're " + str(age) + ".\n")
birthday_wishes2(age= 10)
```

Global variables are accessible from everywhere
Local variables are only accessible in their block
Scope is the area of the code in which the variable is defined

Scope and functions

```
def func1():
    variable1 = 1
```
local variable scope func1

```
def func2():
    variable2 = 2
```
local variable scope func2

```
variable0 = 0
```
global variable global scope

Files

Reading - do not use, good to know

```
#Open and Read File
my_file = open("read_it.txt", "r")
#Close File after using it
my_file.close()
```

Access modes

`"r"/"r+"` Read from / Read from and write to.
Error if the file doesn't exist

`"w"/"w+"` Write to / Write to and read from.
Overwrites if file exists, creates a new file otherwise

`"a"/"a+"` Append / Append and read from
Appends or creates a new file.

Better read - closes file automatically

```
# Make sure you create a workfile.txt
# with random text
with open(read_it.txt', "r") as f:
    read_data = f.read()
print(f.closed) # to show it is closed
print(read_data)
```

Reading characters

```
with open("read_it.txt", "r") as my_file:
    print(my_file.read(1))
    print(my_file.read(5))
```

Reading one line at a time

```
with open("read_it.txt", "r") as my_file:
    print(my_file.readline())
    print(my_file.readline())
    print(my_file.readline())
```

Reading all lines into a list

```
with open("read_it.txt", "r") as my_file:
    lines = my_file.readlines()
    print(lines)
    print(len(lines))
    for line in lines:
        print(line)
```

Looping through the file

```
with open("read_it.txt", "r") as my_file:
    for line in my_file:
        print(line)
```

Creating a file

```
with open("read_it.txt", "w") as my_file:
    my_file.write("Line 1\n")
    my_file.write("Line 2\n")
    my_file.write("Line 3\n")
```

Creating a file from string list

```
with open("read_it.txt", "w") as my_file:
    lines = [ "Line 4\n",
              "Line 5\n",
              "Line 6\n"]
    my_file.writelines(lines)
```

Recap - file functions

```
read() # reads entire file
read(size) # reads that many characters
readlines() # reads lines in a list
readline() # reads chars to end of line
write(output) # writes output to file
writelines(output) # writes strings in
# the list output to a file
```

Exceptions

Examples
(ValueError, ZeroDivisionError, NameError, TypeError)

```
float("Hi!")
10 * (1/0)
4 + spam*3
'2' + 2
```

Handling exceptions

```python
try:
    x = int(input("Type a letter: "))
except(ValueError):
    print("Oops!")
```

Skipping exceptions

```python
try:
    x = int(input("Type a letter: "))
except(ValueError, RuntimeError):
    pass
```

For more exception Types:
docs.python.org/3/library/exceptions.html

Json files (JavaScript Object Notation)

Example json file (**data.json**)

```
{ "name":"Jill",
  "age":31,
  "city":"New York",
  "cats" : [ "jixy",
             "mixy"]}
```

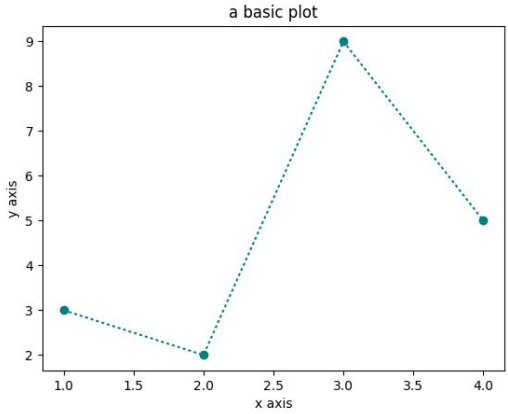
```python
import json
from pprint import pprint
with open('data.json') as data_file:
   data = json.load(data_file)
pprint(data)
print(data["age"])
print(data["name"])
print(data["cats"][0])
```

If it cannot find **pprint** or **json** run

```
pip install pprint json
```

Basic plots

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4]
y = [3, 2, 9, 5]
plt.plot(x, y, 'o:', color='teal')
plt.xlabel('x axis')
plt.ylabel('y axis')
plt.title('a basic plot')
plt.show()
```



If it cannot find **matplotlib**, **urllib** or **pprint** run

```
pip install matplotlib urllib pprint
```

APIs

```
import urllib.request, json
from pprint import pprint
l = "http://weathers.co/api.php?city=Stockholm"
with urllib.request.urlopen(l) as url:
    data = json.loads(url.read().decode())
    pprint(data)
```

Content credits

Mariana Bocoi - mariana.bocoi@gmail.com
Liza Sigova - elizavetasigova@gmail.com
Nadia Mohedano Troyano - nadiamt@gmail.com
Sara Ekman - sara@pinkprogramming.se

Image credits

terminal by Rohith M S
from the Noun Project

PY File by Arthur Shlain
from the Noun Project

remix of
Arrow by Alena Artemova
from the Noun Project

output by Hea Poh Lin
from the Noun Project

Input by Hea Poh Lin
from the Noun Project

Bright Lightbulb by Martyn Jasinski
from the Noun Project

coding by Wilson Joseph
from the Noun Project