Intro to programming - 1

Henri Vandendriessche

21/09/2021

Context

- Why programming ?
- Why python ?
- Why programming in python for cognitive sciences ?

Survival rules for programming

1- Try by yourself before looking for solutions

2- Internet is your best friend

3- Read the manual

4- There is always a manual

5- Have you read the fucking manual ?

6- Not yet ? Then read it

7- Always read the error message

Resources

Books & ebooks

- Gérard Swinnen Apprendre à Programmer avec Python 3 (5e edition) http://inforef.be/swi/python.htm
- Al Sweigart How to automate the boring stuff with Python (2e edition) https://automatetheboringstuff.com/
- Al Sweigart Invent Your Own Computer Games with Python (4e edition) http://inventwithpython.com/invent4thed/

Online course & Mooc

- openclassrooms: - https://openclassrooms.com/fr/courses/7168871-apprenez-les-bases-du-langage-python - https://openclassrooms.com/en/courses/6902811-learn-python-basics - ...

-https://www.my-mooc.com/fr/mooc/python-des-fondamentaux-a-lutilisation-du-langage/

Websites

Writing and running a program with python

- Open sublime
- Write: `$ print("Hello !")`
- Save the file as *hello.py*
- Open a terminal and navigate to your program and run the command: `$ python hello.py`

Variables and data in python

- Manipulating and storing data
- Declaring variables
- Modifiying variables
- Naming variables: - Use alphanumeric characters and underscores (but no number nor underscore as first character) - Use a descriptive name - Avoid short versions - Variable are case-sensitive: upper-case and lower-case make a difference
- Different types of variables: Primitive types
  - intergers
  - float
  - string
  - boolean

Numeric data et operations 1/2

- Differences between integers and floats
- Arithmetic operators + - * / %
- Exercice : calculate and print the result of this operation

$$\frac{15}{3+2} - (\frac{\frac{100}{4}}{5}) * 2$$

Numeric data et operations 2/2

- Differences between integers and floats
- Arithmetic operators + - * / %
- Exercice : calculate and print the result of this operation

$$\frac{15}{3+2} - (\frac{\frac{100}{4}}{5}) * 2$$

```
x = 15/(3+2) - (100/4/5)*2
print(x)
```

```
## -7.0
```

Strings 1/3

- Used to store text (most of the time)
- Strings are declared with " " or ' '. Examples: `$ animal = "Dog" $ class = "Intro to programming in python"`
- String can easily be concatenated the operator +

```
greetings = "Hello"
presentation = "My name is"
name = "Henri"
print( greetings + presentation + name)
```

```
## HelloMy name isHenri
```

Strings 2/3

- Used to store text (most of the time)
- Strings are declared with " " or ' '. Examples: `$ animal = "Dog" $ class = "Intro to programming in python"`
- String can easily be concatenated the operator +

```
greetings = "Hello"
presentation = "My name is"
name = "Henri"
print( greetings + presentation + name)
```

```
## HelloMy name isHenri
```

- Exercice: insert whitespace between the words.

Strings 2/3

- Used to store text (most of the time)
- Strings are declared with " " or ' '. Examples: `$ animal = "Dog" $ class = "Intro to programming in python"`
- String can easily be concatenated the operator +

```
greetings = "Hello"
presentation = "My name is"
name = "Henri"
print( greetings + presentation + name)
```

```
## HelloMy name isHenri
```

- Exercice: insert whitespace between the words.

```
greetings = "Hello"
presentation = "My name is"
name = "Henri"
print( greetings + ' ' + presentation + ' ' + name)
```

Strings 3/3

- Functions working with strings:
  - **len()** calculate the length of a function Example

```
greetings = "Hello"
print(len(greetings))
```

```
## 5
```

  - **replace()** can replace a pattern in a string Example

```
greetings = "Hello"
print(greetings)
```

```
## Hello
```

```
print(greetings.replace('e','a'))
```

```
## Hallo
```

- ...

Booleans

- A boolean is a pretty simple variable but sometimes complex to deal with
- It can have only two values **True** or **False**
- It is used usually to store the truth value of logic

Example: `$ Is_it_Weekend = False $`

```
print(Is_it_Weekend)

## False
```

Program flow 1/4

- A program normally executes sequentially from top to bottom
- But some instructions can change the flow. For examples *if* and *for* loops
- **if** will test for a conditional comparison. If a condition is true then we can execute some lines of codes if it's false we will not execute those lines of codes (but why not others).

Example 1:

```
$   Weekend = "False" $   if Weekend : $
print("Time to go to work")
```

Program flow 1/4

- A program normally executes sequentially from top to bottom
- But some instructions can change the flow. For examples *if* and *for* loops
- **if** will test for a conditional comparison. If a condition is true then we can execute some lines of codes if it's false we will not execute those lines of codes (but why not others).

Example 1:

```
$   Weekend = "False" $   if Weekend : $
print("Let's do nothing and chill") $   else: $
print("Time to go to work")print("Let's do nothing
and chill"
```

Program flow 1/4

- A program normally executes sequentially from top to bottom
- But some instructions can change the flow. For examples *if* and *for* loops
- **if** will test for a conditional comparison. If a condition is true then we can execute some lines of codes if it's false we will not execute those lines of codes (but why not others).

Example 1:

```
$   Weekend = "False" $   if Weekend : $
print("Time to go to work") $   else:" $
print("Let's do nothing and chill"
```

Example2 :

```
$   string = "This string is long but not that long.
How long is it ?" $   if len(string) < 10: $
print("This string has less than 10 character") $
elif len(string) < 20: $       print("This string has
less than 20 character") $   else len(string) >30: $
print("That string is way too long for me   ")
```

Program flow 1/4

- A program normally executes sequentially from top to bottom
- But some instructions can change the flow. For examples *if* and *for* loops
- **if** will test for a conditional comparison. If a condition is true then we can execute some lines of codes if it's false we will not execute those lines of codes (but why not others).

Example 1:

```
$   Weekend = "False" $   if Weekend : $
print("Time to go to work") $   else:" $
print("Let's do nothing and chill"
```

Example2 :

```
$   string = "This string is long but not that long.
How long is it ?" $   if len(string) < 10: $
print("This string has less than 10 character") $
elif len(string) < 20: $       print("This string has
less than 20 character") $   else : $
print("That string is way too long for me   ")
```

Program flow 2/4

- You can also define other alternatives with **elif**
- you can make use different ways to make your comparison: - **and** - **or** - **not**
- You can as well use comparative expressions: - < - > - <= - == tests for equality - != tests for inequality

Be careful = is not the same as ==

```
age = 30

age == 30
```

```
## True
```

Program flow 3/4

- **for** loops iterate one or a set of operations several times.

Example 1:

```
$  for x in range(10): $     print(x)
```

```
for x in range(10):
  print(x)
```

```
## 0
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
```

Program flow 3/4

- **for** loops iterate one or a set of operations several times.

Example 1:

```
$  for x in range(10): $     print(x)
```

```
for x in range(10):
  print(x)
```

```
## 0
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
```

Program flow 4/4

- With **for** loops you see that your program flow is not always unilateral from the top to the bottom of your script
- Let's play a small game. Download

https://github.com/chrplr/PCBS/blob/master/games/human-guess-a-number.py

Program flow 4/4

- With **for** loops you see that your program flow is not always unilateral from the top to the bottom of your script
- Let's play a small game. Download

https://github.com/chrplr/PCBS/blob/master/games/human-guess-a-number.py

- Then go on http://pythontutor.com/ and paste the code of the game.
- Look at the program flow to see how it jumps from one code section to another

Excercices

- Exercice 1: Write code that prints the string “All work no play makes Jack a dull boy” 50 times
- Exercice 2: Write code that prints the squares of all integers between 1 and 100 using range
- Exercice 3: Write code that browses the integer from 0 to 100 but only prints the number 1, 50 and 100
- Exercice 4: Write code that prints only odd numbers between 0 and 100
- Exercice 5: Write code that computes the factorial of an integer (no function, no recursion, just a loop)