Durham University

Learning to Programme with Python

Instructors:
Dmitry Nikolaenko, Niklas Ruth, Jiada Tu
(Advanced Research Computing)

June 13, 2024

Outline

Introduction

Part I

- Installing python interpreter and environment/editor
- Basics – variables, datatypes, comments, built-in functions and math operators
- Getting data in and out – user input, reading and writing files

Part II

- Lists and repetitions
- Conditionals and flow controls
- Functions

Part II

Yet another IDE: Sublime Text

- https://www.sublimetext.com/
- Download
- Install
- Write python scripts in it
- Run python scripts

Durham University

Lists and repetitions

Lists

- A list contains items separated by commas and enclosed with square brackets `[]`
- A list can contain different datatypes (*int*, *float*, *string*, etc.)
- Similar to strings, list members can be accessed using their associated index which is zero-based
- Common list functions: `append()`, `remove()`, `pop()`

```
#examples
list1 = [1, 'A', 4.6]
list2 = [1, 2, 5]
print(len(list1))
>3
print(list1[1:])
>['A', 4.6]
print(list1 + list2)
>[1, 'A', 4.6, 1, 2, 5]
```

For-loops

for *variable name* **in range**(*number of times to repeat*):

statements to be repeated

```
for i in range(10):
  print( 'Hello ')
  print(i) # what do you think is the output?
  print(i+1, '-- Hello ')
```

`For` loop

- A `for` loop is a control instruction used for iteration and repetition
- **iteration** .. perform same operation on different items, one item at a time
- **repetition** .. perform an operation more than once
- Keywords: `for`, `in`
- `i` .. loop control variable or iteration variable
- The colon ':' indicates where the body of `for` loop begins (with *indentation*)
- `range()` function creates a list of integers

```
my_list = ["Andrew", "John", "Kate"]
for i in my_list:
  print(i)
for item in my_list:
  print(item)
print("All items are printed")
for i in range(3):
  print("item", i)
for i in range(1,4):
  print("item", i)
for i in range(2,15,3):
  print("item", i)
```

The `range` function

Statement	Values generated
range(10)	0,1,2,3,4,5,6,7,8,9
range(1,10)	1,2,3,4,5,6,7,8,9
range(3,7)	3,4,5,6
range(2,15,3)	2,5,8,11,14
range(9,2,-1)	9,8,7,6,5,4,3

`While` loop

- A `while` loop is a control instruction ideal for indefinite loops when we don't know when it ends
- Keywords: `while`
- `n=1` .. initialisation
- `n<=10` .. check a condition
- `n+=1` .. update
- Again, the colon ':' indicates where the body of `while`-loop begins (with *indentation*)

```
n=1
while n<=10:
  print(n)
  n+=1
items = 15
i = 2
while i<=items:
  print("item", i)
  i+=3
```

Durham University

Conditionals and flow controls

If statements

`if` (*condition*):

    statements

`else`:

    statements

```
if grade>=80 and grade<90:
  print( 'Your grade is a B. ')
if score>1000 or time>20:
  print( 'Game over. ')
if not (score>1000 or time>20):
  print( 'Game continues. ')
```

Conditional operators

Expression	Description
`if x>3:`	if x is greater than 3
`if x>=3:`	if x is greater than or equal to 3
`if x==3:`	if x is 3
`if x!=3:`	if x is not 3

Common mistakes:

Incorrect	Correct
`if x=1:`	`if x==1:`

Common mistakes

Incorrect	Correct
`if x=1:`	`if x==1:`
`if x>1 or x<100:`	`if x>1 and x<100:`
`if grade>=80 and <90:`	`if grade>=80 and grade<90:`

Elif statement

```
grade = eval(input( 'Enter your score: '))
```

Using only `if`:

```
if grade>=90:
  print( 'A ')
if grade>=80 and grade<90:
  print( 'B ')
if grade>=70 and grade<80:
  print( 'C ')
if grade>=60 and grade<70:
  print( 'D ')
if grade<60:
  print( 'F ')
```

Using `if` and `elif`:

```
if grade>=90:
  print( 'A ')
elif grade>=80:
  print( 'B ')
elif grade>=70:
  print( 'C '):
elif grade>=60:
  print( 'D '):
else:
  print( 'F ')
```

Exercises – for loop

1. Write a program that prints your name 100 times.
2. Write a program to fill the screen horizontally and vertically with your name. [Hint: add the option end='' into the print function to fill the screen horizontally.]
3. Write a program that outputs 100 lines, numbered 1 to 100, each with your name on it. The output should look like the output on the right -->:
4. Write a program that uses a for loop to print the numbers 100, 98, 96, . . . , 4, 2.
5. Write a program that uses exactly four for loops to print the sequence of letters below:

   AAAAAAAAAABBBBBBBCDCDCDCDEFFFFFFG

```
1 Your name
2 Your name
3 Your name
4 Your name
...
100 Your name
```

Exercises – if statement

1. Write a program that asks the user to enter a length in centimeters. If the user enters a negative length, the program should tell the user that the entry is invalid. Otherwise, the program should convert the length to inches and print out the result. There are 2.54 centimeters in an inch.

2. Ask the user to enter a temperature in Celsius. The program should print a message based on the temperature:
    - If the temperature is less than -273.15, print that the temperature is invalid because it is below absolute zero.
    - If it is exactly -273.15, print that the temperature is absolute 0.
    - If the temperature is between -273.15 and 0, print that the temperature is below freezing.
    - If it is 0, print that the temperature is at the freezing point.
    - If it is between 0 and 100, print that the temperature is in the normal range.
    - If it is 100, print that the temperature is at the boiling point.
    - If it is above 100, print that the temperature is above the boiling point.

Durham University

Functions

Function basics

- Function definition
- Arguments
- Return statement
- Function call
- Scope of variables

```
def print_hello():
  print( 'Hello! ')

print_hello()
print( '1234567 ')
print_hello()
```

```
def convert(t):
  return t*9/5+32

print(convert(20))
```

Functions

- Default arguments
- Local variables

```
def func1(int n=10):
  for i in range(n):
    print(i)

def func2():
  i=100
  func1()
  print(i)
```

Exercises - functions

1. Think about how to best name the variables and functions in the examples above, and why meaningful names are crucial.
2. Write your own functions, and let one be called from within the other.
3. Write a code that uses all that you have learned today

Next: Coding with Python (intermediate level)

- When: Monday, 24th of June
- ‘Pythonic’ concepts
  - List comprehension, ternary expressions, *args and **kwargs
  - Lambda functions
- More advanced string manipulation
- Introduction to libraries and modules
  - Module csv for writing/reading csv files
- Data structures and containers
  - Dictionaries, sets, tuples
- Brief introduction into classes

Durham University

Thank you for your attendance!

- Feedback would really be appreciated: https://bit.ly/arc_trainingfeedback2024
- Other training courses at ARC
- RSE support