***Learn Python The Hard Way***

**Top-down coding**

1. Write or draw about the problem.
2. Extract key concepts from 1 and research them.
3. Create a class hierarchy and object map for the concepts.
4. Code the classes and a test to run them.
5. Repeat and refine.

**Bottom-up coding**

1. Take a small piece of the problem; hack on some code and get it to run barely.
2. Refine the code into something more formal with classes and automated tests.
3. Extract the key concepts you’re using and research them.
4. Write a description of what’s really going on.
5. Go back and refine the code, possibly throwing it out and starting over.
6. Repeat, moving on to some other piece of the problem.

|  |  |
| --- | --- |
| **Code** | **Functie** |
| @ |  |
| # | wordt gebruikt om commentaar toe te voegen, dat niet gezien wordt als code door python |
| [:] |  |
| <, >, <=, >= | controleert of het linker lid:   * kleiner dan, * groter dan, * kleiner dan of gelijk aan, * groter dan of gelijk aan   het rechter lid is |
| “““ … ””” | laat het schrijven van *documentation comments* toe, die niet gezien worden als code door python |
| “ … ” \* n | schrijft een *string* n keer achter elkaar |
| .append() | voegt een element toe aan een lijst |
| argv | maakt het mogelijk argumenten toe te voegen bij het uitvoeren van een script |
| assert |  |
| b‘ ’ | geeft aan dat een reeks tekens een *byte string* is |
| boolean expressions |  |
| break | beëindigt de huidige *loop* |
| .capitalize() | maakt van het eerste teken in de *string* een hoofdletter en van de rest kleine letters. |
| class x: | maakt een nieuwe klasse met naam “x” aan |
| .close() | sluit een bestand |
| continue | beëindigt de huidige *iteration* van een *loop* |
| .count() |  |
| .decode() | zet een *byte string* om in een gewone *string* (via UTF-8 bijvoorbeeld) |
| def x(): | maakt een nieuwe functie met naam “x” aan |
| del (delete) | verwijdert elementen uit een *iterable* |
| dict (dictionary) {} | maakt een nieuwe *dictionary* aan; bijvoorbeeld: talen = {‘Nederlands’: ‘huis’, ‘English’: ‘house’, ‘Deutsch’: ‘Haus’} |
| elif *test expression*: | maakt een ondergeschikte *conditional* aan; voert een blok code alleen uit als de *test expression* waar is |
| else: | voert een blok code alleen uit als de voorgaande *if-* en *elif-statements* niet uitgevoerd worden |
| .encode() | zet een *string* om in bytes (via UTF- 8 bijvoorbeeld) |
| end= | bepaalt waarmee een regel eindigt |
| error (type) |  |
| escape |  |
| except |  |
| exec |  |
| exists() | controleert of een map of bestand bestaat; heeft als output *True* of *False* |
| exit() |  |
| finally |  |
| float() | zet een *string* met cijfertekens om in een kommagetal (*floating point number*) |
| for … in … |  |
| for … in range(n, m, p) | een *loop* die (m–n)/p keer uitgevoerd wordt; n is de beginwaarde, m de eindwaarde en p de stapgrootte |
| formatting |  |
| .get(‘x’) | haalt uit een *dictionary* de waarde die is opgeslagen onder de ‘x’ *key* |
| global *var* |  |
| if *test expression*: | maakt een *conditional* aan; voert een blok code alleen uit als de *test expression* waar is |
| import …, import … from … | laadt modules in python in |
| … in … |  |
| \_init\_() (*initialize*) |  |
| input() | laat de gebruiker informatie invoeren |
| int() | zet een *string* met cijfertekens om in een geheel getal; rondt naar beneden af als dat nodig is |
| is |  |
| .isnumeric() | controleert of een *string* alleen uit cijfertekens bestaat; heeft als output *True* of *False* |
| .items() | geeft de inhoud van een *dictionary* weer in de vorm van een verzameling |
| .join() | verzamelt alle elementen van een *iterable* bijeen; deze worden gescheiden van elkaar met de meegegeven *string separator* |
| len() | telt uit hoeveel tekens een *string* bestaat; heeft als output dat aantal |
| lambda |  |
| None |  |
| open() | opent een bestand in Python als een *file object* |
| pass | vertelt python een stuk code uit te voeren als niks |
| pop() | snijdt het n-de element af van een lijst; dat getal heet de *index* en die heeft als beginwaarde 0 |
| print(“““ … ”””) | maakt het mogelijk om meerdere regels tegelijk te printen |
| raise |  |
| .read() | leest een bestand in |
| .readline() | leest een bepaalde regel van een bestand in |
| .replace() |  |
| return | geeft iets als de output van een functie, bijvoorbeeld een waarde die aan een variabele wordt toegekend |
| round(n, m) | rondt een getal *n* af op *m* decimalen |
| .seek() |  |
| sorted() | sorteert de elementen van een lijst op- of aflopend; het resultaat is een nieuwe lijst |
| .split() | splitst een *string* op in delen, die de elementen worden van een nieuwe lijst; tenzij aangegeven, vindt de splitsing plaats op witruimte |
| .strip() | verwijdert witruimte en eventueel andere tekens uit de *string* |
| super() |  |
| superclass, subclass |  |
| .truncate() | leegt een bestand |
| try |  |
| UTF-8 | *Unicode Transformation Format 8 bits* – standaard die door python gebruikt wordt om tekens aan een getal te koppelen |
| var = *waarde* | slaat een variabele op met een bepaalde waarde |
| while *test expression*: | voert achter elkaar door een blok code uit zolang als de *test expression* waar is |
| with |  |
| with … as … |  |
| .write() | schrijft tekst vanuit python naar een bestand |
| yield |  |
|  |  |

De *escape sequences* van Python

|  |  |
| --- | --- |
| **Input** | **Function** |
| \\ | Backslash |
| \‘ | Single-quote |
| \“ | Double-quote |
| \a | ASCII bell |
| \b | ASCII backspace |
| \f | ASCII formfeed |
| \n | ASCII linefeed |
| \N{name} | Character named *name* in the Unicode database |
| \r | Carriage return |
| \t | Horizontal tab |
| \uxxxx | Character with 16-bit hex value xxxx |
| \Uxxxxxxxx | Character with 32-bit hex value xxxxxxxx |
| \v | ASCII vertical tab |
| \000 | Character with octal value 000 |
| \xhh | Character with hex value hh |

Ideeën voor een Python script

* *choose function*
* spel
* decimalen rationele getallen
* twin primes