16 Aug '24

(1) PyCharm - application - to do Python programming
(Community edition)
- Editor, File tree, Output, Console

(2) Book : Learning Programming with Python 3.

(3) Basic expressions - integers, floats, strings
datatypes

HW: Finish last class HW
- 3.expressions file also

21 Aug'24

Datatypes — integers, floats, strings

23, 42, -64 23.1, 43.0, -61.7 "This" 'ant'

Expressions — simple calculations

→ 3+4+5 (int)

→ 42.6+ 53.2 (float)

→ 43 + 53.2 (float — int is converted to float)

→ 'this' + 'is' (string)

→ '3+4=' + str(3+4)

```
>>> type(32)          → int
>>> type(23.4)        → float
>>> type('vaptx')     → str
>>> type("vaptx")     → str
```

```
>>> type(vaptx)
→ error
(no quotes)
```

Operators :

+ → addition 3+4 = 7

− → subtraction 10−4 = 6

* → multiplication 20*4 = 80
(asterisk)

/ → division 20/4 = 5
(slash)

21/4 = 5.25

// → int. division 21//4 = 5
(quotient)

% → modulo 21%4 = 1
(remainder)

** → power 4**6 = 4^6

= 4×4×4×4×4×4
(6 times)

4*6 = 4×6 = 4+4+4+4+4

4**6 = 4^6 = 4×4×4×4×4×4

Empty space:

Same → 3+4 , 3. + 4 , 3 + 4

Type casting:

int → float : $\underset{int}{30} + \underset{float}{42.3} = \underset{float}{30.0} + \underset{float}{42.3} = 72.3$

float → int : 30 + int(42.3) = 30 + 42 = 72

int → str : $\underset{str}{\text{"Result is"}} +_{str} (\underset{int}{42*63})$

= "Result is" + '2709'

= Result is 2709

str → int : int('24') = 24 ; int('3c4k') ⟹ error

str → float : float('24.3') = 24.3 ;

String operators:

3 * 'Pig' = 'PigPigPig'

'Pig' + 'Car' = 'PigCar'

HW: Ch. 3 textbook

→ Create a program file

→ Type everything that is in boxes in the text book

→ Run

→ Play with operators, expressions, type casting

→ From Maths class HW +, -, *, / (10 for each) do all of those in python and verify (create a program called Math HW)

09 Sep'24

Operator precedence (order):

$3 + 4 - 6 * 14 / 2 + 3*(4-1)$

float (24)

(1)

$4 + 4 / 2$

$= 8/2$ ✗

$= 4$

(2)

$4 + 4 / 2$

$= 4 + 2$ ✓

$= 6$

Maths

BODMAS — Brackets, of, division, multiply, add, subtract

PEMDAS — parantheses, exponential, multiplication,
( ) (power)
division, add, subtract

Python

Variables: A place where you store values.

You can give any name to a variable.

Ex: myVar = 43

→ myVar + 23

66

→ myVar * 2

86

x = < some value >

↳ assignment operator

Variable names cannot by python keywords

— Eg: for, in, true, false etc.

HW: (1) Read all previous class notes

(2) Ch. 4 code type into a program file and run

19 Sep' 24

Ch.4 variables :

→ Can u name a variable as 'print' ?

Yes. But after that, regular print function will not work

→ How to block comment ?

select all the lines you want to comment and press 'ctrl + /'

— same for uncommenting

→ HW: (1) Swapping examples practice - write code

→ Jim has 23.56 grams of tomatoes, while Tom has 43.641 grams of beans. They exchange their vegetables. Do this in python as shown in class example.

→ $x =$ 'expr'

↳ First this expression is computed

→ Then 'x' variable is assigned that computed value.

$x = 2$;

$y = x*5 + 2$;

(Here 'x' is 2, 'y' is 12)

$x = x + y*2$

$2 + 12*2$

$2 + 24$

26

PEMDAS-

After this step, x is '26', y is '12'

$y = x - 12$;

$26 - 12$

14

After this step, x is 26, y is '14'.

→ **HW:** After each of these steps, manually compute the values of all variables; Check in python also by printing the values.

$x = 2$; $y = 3$

(1) $z = x^2 + y$;

(2) $y = x - y$;

(3) $x = z * y + x - 14$;

(4) $z = y^2 - 15$;

(5) $x = x + y - z$;

25 Sep '24

For Maths arithmetics - +, -, /, *,
we learnt how to verify
'==' operator - equality operator

```
print('expr1' == 'expr 2')
```

print true if expr1 and 2 are equal
else prints false

→ sum = 23+42

```
print( sum - 23 == 42)
```

→ product = 23*42

```
print( product // 42 == 23 )
```

→ difference = 23 - 42
print (difference + 42 == 23)

→ quotient = 231 // 53 ; remainder = 231 % 53
print (quotient * 53 + remainder == 231)

<u>HW:</u> For previous maths arithmetics HW, verify each answer like in class example code.

Add, Multiplications	— , ÷
Verify in two ways	only one way to verify
sum = a+b; sum - a == b sum - b == a	difference = a-b; difference+b == a
product = a*b; product/a == b product/b == a	dividend // divisor = quotient ; quotient*divisor + remainder == dividend dividend % divisor = remainder

01 Oct '24

Boolean data type:

⟹ Has only two values 'true' or '-false'

Integers can have infinitely diff- values → -43, -36, 0, 46, 53 . . .

Floats " " " " " → -2.3, -7.9, 796.432 . . .

→ Equality operator results in a boolean value (expr1 == expr2) is true or false

→ You can create boolean variables

myBoolean = (expr1 == expr2)

→ == equality operator

(expr1 == expr2) gives 'true' if they are equal
gives 'false' if not equal

→ != (not equal operator)

(expr1 != expr2) gives 'true' if they are not equal
gives 'false' if they are equal.

→ HW: For the previous Arithmetic operations, use variable names for all values - like in the addition example from class. Copy code for all other additions. Same for -, *, /.
Give examples for '==' & '!='

→ Briefly looked at AI to summarize class material, notes, texts etc.

---

10 Oct '24

special characters '\n' → newline
'\t' → tab

You can use them in strings

```
>>print ("abc\n xyz\t pqr\t stu\n"); print ('123');
```

Output:

```
abc
xyz     pqr     stu

123
```

⇒ Arrays, Loops, Functions

Function — A few lines of code that you can reuse many times.

Eg :

function name

parameters

define keyword

```
def addition( var1 , var2 )
    sum = var1 + var2
    print(sum)
    return sum
```

indentation (tab space)

p = addition(23, 56)

↳ This is called 'function call' or 'calling the function' 'addition'

HW:

→ Look at 'addition' function from class and write similar functions for subtraction, multiplication, division

→ Add examples for function calls also